

Zabbix系统
中文组成员
力作

Zabbix系统源码级功能增减功力及15年运维经验体现

Zabbix 监控系统

王余应 著

Zabbix系统配置、分布式监控管理、优化及常见问题梳理
案例典型、来源于实际监测应用，工作中可以直接借鉴



中国工信出版集团



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
<http://www.phei.com.cn>

Zabbix 监控系统

王余应 著

電子工業出版社.

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书共分为9章,介绍了Zabbix系统的体系结构、安装配置方法、数据采集方法、各个模块的应用和配置方法,以及Zabbix系统的基本原理和规则等,并从操作系统层面、数据库层面和Zabbix系统组件层面介绍了Zabbix系统的优化方法。最后,作者结合多年的运维实践经验,总结出维护和管理Zabbix系统过程中所遇到的常见问题和技巧。本书是作者多年来实战经验的总结和浓缩,全书在讲解过程中也穿插介绍了与系统监控相关的周边知识,以及其在实际应用中的操作。

本书在文字叙述上力求条理清晰、通俗易懂,并提供了大量的完整实例和代码,适合系统监控工程师、运维工程师、监控和运维自动化系统开发工程师、系统调优师、应用系统测试人员,以及监控/运维自动化系统的系统架构师等阅读;对于大中专院校的教师、学生,运维团队的技术负责人,以及其他对系统监控感兴趣的读者,本书也具有非常高的阅读价值。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有,侵权必究。

图书在版编目(CIP)数据

Zabbix 监控系统 / 王余应著. —北京:电子工业出版社, 2015.5

ISBN 978-7-121-25682-0

I. ①Z… II. ①王… III. ①计算机监控系统 IV. ①TP277

中国版本图书馆CIP数据核字(2015)第047742号

责任编辑:张月萍

特约编辑:梁卫红

印 刷:三河市鑫金马印装有限公司

装 订:三河市鑫金马印装有限公司

出版发行:电子工业出版社

北京市海淀区万寿路173信箱

邮编:100036

开 本:787×1092 1/16

印张:22.25

字数:583千字

版 次:2015年5月第1版

印 次:2015年5月第1次印刷

印 数:3500册

定价:59.00元

凡所购买电子工业出版社图书有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系,联系及邮购电话:(010)88254888。

质量投诉请发邮件至 zltts@phei.com.cn, 盗版侵权举报请发邮件到 dbqq@phei.com.cn。

服务热线:(010)88258888。

推荐序

当前世界是风起“云”（计算）涌的时代，SaaS（软件即服务）改变了传统软件实施的方式，而 PaaS（平台即服务）则进一步改善了各种应用系统的生态环境；“物联网”全面走入我们生活的角角落落只是时间问题；“智能社会”的来临正随着全社会网络基础架构的改善而一点点呈现在人们面前。

这些，在广泛降低了用户端软硬件投入的基础上，却对集中式的服务器端/群等提出了更为严峻的考验。怎样才能保障系统工作的高效性、稳定性、可靠性？怎样才能动态地了解用户的需求和分配资源给到用户？怎样才能应对潜在的网络中的各种风险？这些都需要能够及时发现并采取有效措施及时解决，所以系统监控工作无疑占据了日常 IT 运维中非常重要的地位。

一个人干不过一个团队，一个团队干不过一个系统。有一套好的监控系统软件，可以让你高枕无忧。

本书所介绍的这款开源监控系统——Zabbix 监控系统，它不但功能强大、输出的数据图形美观，而且操作和管理都非常简单方便。监控系统作为一类专业性较强的应用系统，它与普通应用软件还是有着非常大的差别的。要理解和掌握一款监控系统，仅仅熟练地掌握其日常操作是远远不够的，还需要掌握大量的周边知识，例如网络知识、操作系统知识、数据库知识、编程知识，等等。

我与本书的作者——王余应相识于 15 年前，那时候互联网在中国还是一个十分新鲜的事物，王余应作为我团队里的一员骨干力量，从那时起即担当公司服务器的日常监控和管理工作。至此他在系统管理和监控这一领域至少摸爬滚打了十几年。十多年运维的从业经历，使他积累了非常丰富的一线运维经验和宽广的知识面，使他不仅能将 Zabbix 这一开源的监控系统讲解透彻，而且还穿插讲解了许多与系统监控有关的周边知识，而这也是本书的一大特色。

本书从最基础的 Zabbix 系统安装部署讲起，直至 Zabbix 系统的优化，包括了日常管理和维护 Zabbix 系统的各个方面，是一本不可多得的介绍 Zabbix 监控系统的专业书籍。在体系安排上，本书遵循由易到难、循序渐进的原则；在行文上，叙述语言通俗易懂、结构清晰，特别是本书提供了较多的实例，这些实例可以为读者实际工作提供较高的借鉴价值。

CareFusion 亚太 IT 总监, 汤国忠

2015 年 2 月

前言

由此可见，健壮的监控系统会在运维工作中发挥着十分重要的作用。它不仅能大大提升运维工作的效率，减少工作失误，使运维人员能够及时发现系统所出现的故障和问题，而且可以通过对监控数据的分析，找出系统性能瓶颈，为系统架构的重构提供数据支撑。

在笔者所接触的开源监控系统中，只有 Zabbix 系统才能算得上是真正意义上的企业级开源的分布式监控系统。它所具有的丰富的数据采集方法，使它可以采集和处理几乎所有类型的监控数据；而它所具有的灵活的报警机制，使它可以实现智能、灵活的报警策略；其 Web 组件不但方便我们日常管理和维护，而且可以输出近乎完美的数据图、拓扑图和各种报表；而它所特有的网络自动发现和低级自动发现功能，不仅能大大提高我们的工作效率，减少人为出错的可能，且使 Zabbix 系统相比其他开源的监控系统具有更高的“智能”。

在本书中，笔者不但尝试以通俗易懂、结构清晰的语言向读者介绍 Zabbix 系统中各种强大和灵活的功能，使读者能够即学即用，以此来节约读者宝贵的时间；而且在本书的最后一章，笔者还将自己在管理和使用 Zabbix 系统的过程中所遇到的常见问题，进行了总结和整理，以期能使读者避免笔者所走过的弯路。

最后，如果你也想见识一下 Zabbix 系统所具有的强大、灵活和智能的监控功能，那么请从本书的第 1 章开始吧！

本书特色

1. 内容全面，本书不但对 Zabbix 系统各个功能做了详细的介绍，而且较详细地介绍了与监控系统相关的周边知识。
2. 结构清晰，全书整体结构上遵循从易到难的顺序，且各章节之间都有较强的连续性。

3. 实用性强, 本书安排了大量的实例, 且这些实例均来自于笔者在工作中的实际监控应用。因此, 本书所阐述的实例, 读者几乎可以不做修改或稍做修改就可以运用到自己的实际工作中。
4. 通俗易懂, 本书力求以通俗的语言介绍 Zabbix 系统中强大和灵活的功能。
5. 用语规范, 本书力求对于计算机方面的专业术语应用到位, 严格遵循计算机科学的学术要求。
6. 实例典型, 本书所列举的大量实例中, 力求做到不重复, 且具有典型性。

本书内容及知识体系

第1章 Zabbix 系统介绍

本章首先介绍了什么是 Zabbix 系统, 它的特点以及它与其他开源监控系统比较有什么优缺点等; 之后, 简单地介绍了 Zabbix 系统中各个组件、体系结构及其各个组件之间的关系; 接下来, 详细介绍了如何安装和部署一套 Zabbix 系统; 最后, 对 Zabbix 系统中一些常见的概念做了简单的介绍和说明, 为后续章节做必要的准备。

第2章 数据采集方法介绍

本章逐一介绍 Zabbix 系统中所使用到的多达十多种的监控数据采集方法的原理、相关的周边知识及与之相关的 Zabbix 服务器和客户端配置等做了详细介绍。

第3章 Zabbix 系统配置基础

本章详细介绍了 Zabbix 系统 Web 组件中各种查看类菜单的功能和作用, 以及它们对应页面所显示内容的含义; 并简单介绍了如何在 Zabbix 系统中添加新用户、配置被监控主机、监控项目、消息介质和动作等。

第4章 Zabbix 系统中相关规则及原理

本章不仅介绍了 Zabbix 系统中监控项目关键字的命名格式和规范, 以及系统中所预定义的关键字, 还介绍了时间区间的定义方法、历史数据与趋势数据的联系与区别、什么是数据映射以及被监控设备代理组件的扩展等方面内容; 同时, 还对什么是动态索引、Zabbix 系统的事件和事件源, 以及动作行为的升级过程和原理做了详细的介绍和说明。

第5章 Zabbix 系统配置进阶

本章详细介绍了模板、正则表达式、低级自动发现等方面的内容, 并进一步介绍了触发器表达式、触发器等级、动作的触发条件和行为等方面的内容; 并且, 实际动手创建了多个项目样板、触发器样板和数据图样板等。

第6章 Zabbix 系统高级配置及日常管理

本章介绍了在 Zabbix 系统中创建网络拓扑图、图表和幻灯片的方法和操作过程, 并介绍了如何在 Zabbix 系统中配置主机资产、Zabbix 系统认证方式、脚本, 以及用户、用户组、IT 服务和常规设置等方面的内容; 并在此基础上, 介绍了在使用 Zabbix 系统 Web 前端组件时可能经

常使用到的操作方法和技巧,如批量更新、维护模式配置、配置的导出与导入以及全局搜索等。

第7章 分布式监控

本章详细介绍了 Zabbix 系统所提供的两种分布式解决方案,即单级分布式解决方案和多级分布式解决方案以及它们的配置方法等。

第8章 Zabbix 系统优化

本章首先对 Zabbix 系统的特点做出了分析,从而得出 Zabbix 系统是属于一种偏写入型的重数据库的应用系统。在此基础上,本章讨论了在对 Zabbix 系统进行优化时应遵循的一些原则,并从操作系统、数据库和 Zabbix 系统组件等三个层面讨论了如何针对 Zabbix 系统进行优化。

第9章 常见问题及使用技巧

本章详细讨论了在日常维护和管理 Zabbix 系统的过程中,可能遇到的常见问题,从原理上分析了这些问题产生的原因,并给出了解决方法。

适合阅读本书的读者

- 系统监控工程师
- 运维工程师
- 中/高级 Linux 系统管理员
- 监控/运维自动化系统开发工程师
- 系统调优师
- 应用系统测试人员
- 监控/运维自动化系统的系统架构师
- 大中专院校计算机及相关专业的教师、学生
- 其他对系统监控感兴趣的人员

说明

本书采用的 Zabbix 版本为 2.0 和 2.2 版本,因此对 Zabbix 1.8 版本不做讨论。本书所采用的操作系统以 CentOS 6.x 和 Windows 2003 为主,对于其他操作系统,其配置方法类似,请读者举一反三。

作为一种舶来品,Zabbix 系统的汉化效果目前还不十分完美,而笔者作为 Zabbix 系统中文翻译组成员,也是刚刚加入不久,并在为 Zabbix 系统汉化做持续努力。相对于 Zabbix 2.2 版本,Zabbix 2.0 版本的汉化效果要更加完美一些,因此本书的所有截图均是在 Zabbix 2.0 版本上截图的。因为 Zabbix 系统目前的汉化效果还不是十分完美,因此在本书的叙述中中文术语都尽可能地给出了对应的英文,读者在使用不同版本的 Zabbix 系统时,应加以对照,以免因为同一术语在不同版本中译法的不同而产生迷惑。

作者联系方式

由于经验不足和水平有限，书中难免存在错误和不足不处，恳请广大读者批评指正，也欢迎您将错误和建议发送至我的邮箱 `net_use@bzhzy.com`，期待能够收到您的真挚反馈。

致谢

首先，要感谢的是我的父母，是他们含辛茹苦地将我养育成人，并给予了我非常多的鼓励和支持。

其次，要感谢我的爱人，在撰写本书的过程中，是她无怨无悔地承担了几几乎所有的家务和教育孩子的工作。因为有她无私的付出，才能使我安心地撰写本书。

王余应

2015年2月

目 录

第 1 章 Zabbix 系统介绍	1
1.1 什么是 Zabbix 监控系统	1
1.2 Zabbix 监控系统的特点	2
1.3 常见开源监控系统的比较	2
1.4 Zabbix 系统组件及其体系结构	3
1.4.1 Zabbix 系统组件介绍	4
1.4.2 Zabbix 系统各组件之间的关系	5
1.5 部署 Zabbix 系统的软硬件需求	6
1.5.1 安装 Zabbix 系统的硬件需求	6
1.5.2 安装 Zabbix 系统的软件要求	6
1.5.3 关于 Zabbix 系统数据库大小的计算	8
1.6 独立服务器安装与部署	9
1.6.1 安装前准备	10
1.6.2 LNMP 环境安装	11
1.6.3 Zabbix 系统安装	13
1.6.4 部署 Web 前端组件	19
1.7 Zabbix 系统中的基本定义	24
1.8 本章小结	28
第 2 章 数据采集方法介绍	29
2.1 通过被监控设备代理采集数据	29
2.1.1 被监控设备代理被动工作模式	30
2.1.2 被监控设备代理主动工作模式	30
2.2 简单检查	31
2.3 通过 SNMP 协议采集数据	33
2.3.1 SNMP 协议介绍	34
2.3.2 SNMP 协议版本	35
2.3.3 Linux 系统下 SNMP 服务的安装与配置	37
2.3.4 Zabbix 服务器上的 SNMP 陷入配置	40
2.3.5 Windows 2003 下 SNMP 服务的安装与配置	42
2.3.6 通过 SNMP 协议采集监控数据	45
2.4 Zabbix 系统内部数据采集	47
2.5 Zabbix 陷入	52
2.6 数据聚合	52
2.7 通过脚本采集监控数据	54
2.8 数据库监控	54
2.9 通过 IPMI 代理采集监控数据	55

2.10 通过 SSH 协议采集监控数据	57
2.11 通过 TELNET 协议采集监控数据	59
2.12 通过 JMX 协议采集监控数据	59
2.12.1 被监控主机上 JMX 服务的配置	60
2.12.2 Java 应用程序网关的配置	62
2.13 通过计算的方法采集监控数据	63
2.14 本章小结	64
第 3 章 Zabbix 系统配置基础	65
3.1 用户登录及创建新用户	65
3.1.1 用户登录	65
3.1.2 创建新用户	66
3.2 认识 Web 前端组件页面	70
3.2.1 Web 前端组件页面布局	71
3.2.2 Web 前端组件行为配置	72
3.2.3 Web 前端组件维护模式配置	73
3.3 Zabbix 系统菜单项主要功能	74
3.3.1 “状态统计”菜单项的功能	74
3.3.2 “资产记录”菜单项的功能	94
3.3.3 “系统报告”菜单项的功能	95
3.3.4 “高级配置”菜单项的功能	97
3.4 配置第一台被监控主机	99
3.5 配置监控项目	102
3.6 配置触发器	107
3.7 接收第一条报警信息	110
3.7.1 配置 Email 消息介质	110
3.7.2 配置手机短信消息介质	112
3.7.3 创建新动作	114
3.7.4 接收第一条报警信息	118
3.8 本章小结	119
第 4 章 Zabbix 系统中相关规则及原理	120
4.1 监控项目关键字命名规范	120
4.1.1 监控项目关键字命名规范	120
4.1.2 Zabbix 系统中预定义的关键字	122
4.2 时间区间定义方法	124
4.3 历史数据和趋势数据	125
4.4 被监控设备代理组件的扩展	126
4.5 动态索引	128
4.5.1 动态索引介绍	129
4.5.2 特殊 OID 值	131
4.6 事件和事件源	133
4.6.1 触发器类事件 (Trigger events)	134
4.6.2 自动发现类事件 (Discovery events)	134
4.6.3 被监控设备代理自动注册类事件 (Active agent auto-discovery events)	135

4.6.4	内部事件 (Internal events)	136
4.7	动作行为升级	137
4.8	数据映射	139
4.9	宏 (Macro) 及宏的替换顺序	141
4.10	Zabbix 系统报警流程分析	144
4.11	本章小结	146
第 5 章	Zabbix 系统配置进阶	147
5.1	模板的配置与使用	147
5.1.1	查看模板	149
5.1.2	配置模板	150
5.1.3	关联模板到主机	153
5.2	配置监控项目	154
5.2.1	配置获取主机硬件信息的监控项目	155
5.2.2	配置 Web 端口状态监控项目	156
5.2.3	配置 Nginx 状态数据监控项目	158
5.2.4	配置数据库监控项目	160
5.2.5	配置磁盘读取速率监控项目	161
5.2.6	配置 Tomcat 性能监控项目	163
5.2.7	配置 IPMI 监控项目	164
5.3	正则表达式及低级自动发现规则配置	165
5.3.1	正则表达式介绍	165
5.3.2	正则表达式配置	167
5.3.3	低级自动发现功能	170
5.3.4	配置磁盘分区监控项目	171
5.3.5	配置网卡流量监控项目	178
5.3.6	配置网络端口连接数监控项目	180
5.4	数据图及其配置	183
5.4.1	数据图	183
5.4.2	读懂简单数据图	183
5.4.3	网卡流量数据图配置	185
5.5	触发器配置进阶	188
5.5.1	触发器计算表达式	189
5.5.2	关于触发器依赖	190
5.5.3	关于触发器级别	192
5.5.4	配置磁盘分区空间使用率触发器	192
5.6	动作配置进阶	194
5.6.1	关于动作分类	194
5.6.2	关于动作触发条件	196
5.6.3	配置清理磁盘空间动作	201
5.7	网络自动发现配置	203
5.7.1	网络自动发现功能	203
5.7.2	配置网络自动发现规则	204
5.7.3	配置自动发现动作	206
5.8	Web 监控	208

5.8.1	Web 监控介绍	209
5.8.2	Web 监控配置	211
5.9	本章小结	215
第 6 章	Zabbix 系统高级配置及日常管理	216
6.1	配置网络拓扑图	216
6.1.1	定义网络拓扑图	216
6.1.2	编辑网络拓扑图元素	218
6.2	配置图表和幻灯片	222
6.2.1	配置图表	222
6.2.2	配置幻灯片	225
6.3	配置主机资产信息	226
6.4	配置认证方式和脚本	226
6.4.1	配置认证方式	227
6.4.2	配置脚本	228
6.5	配置用户及用户组	229
6.5.1	用户类型及用户权限	230
6.5.2	配置用户组	231
6.6	配置 IT 服务	233
6.7	“常规”配置	236
6.7.1	“图形界面 (GUI)” 配置	236
6.7.2	“管家 (Housekeeper)” 配置	238
6.7.3	“其他参数 (Other)” 配置	238
6.8	日常管理功能介绍	240
6.8.1	批量更新 (Mass update)	240
6.8.2	维护模式	241
6.8.3	事件确认	244
6.8.4	导出与导入	245
6.8.5	全局搜索	246
6.8.6	配置账号属性	247
6.9	本章小结	248
第 7 章	分布式监控	249
7.1	分布式监控介绍	249
7.2	单级分布式监控	250
7.2.1	Zabbix 服务器代理组件	251
7.2.2	Zabbix 服务器代理组件安装	253
7.2.3	Zabbix 服务器代理组件运行环境配置	254
7.2.4	Zabbix 服务器代理节点的添加及使用	257
7.3	多级分布式监控	258
7.3.1	多级分布式监控的结构	258
7.3.2	多级分布式监控系统的安装与部署	260
7.4	本章小结	262

第 8 章 Zabbix 系统优化	263
8.1 Zabbix 系统特点分析	263
8.2 Zabbix 系统调优原则	264
8.3 操作系统优化	267
8.3.1 I/O 优化	267
8.3.2 Linux 内核参数优化	272
8.3.3 关闭非必要服务	275
8.4 MySQL 数据库优化	275
8.4.1 MySQL 服务器配置优化	276
8.4.2 数据库表分区	280
8.4.3 创建自动维护分区存储过程	282
8.5 Zabbix 系统组件优化	286
8.5.1 Zabbix 服务器配置项说明	286
8.5.2 Zabbix 系统数据流分析	290
8.5.3 Zabbix 系统性能问题表现	291
8.5.4 Zabbix 系统内部状态监控	294
8.6 本章小结	295
第 9 章 常见问题及使用技巧	296
9.1 为什么数据图中的中文显示为乱码	296
9.2 如何完善 Zabbix 系统汉化效果	298
9.2.1 基于 gettext 多语言支持系统的开发流程	298
9.2.2 可移植对象文件格式说明	299
9.2.3 Zabbix 系统汉化效果完善	301
9.3 如何批量添加图表	302
9.3.1 基本功能说明	302
9.3.2 数据表关系分析	303
9.3.3 程序流程分析	306
9.4 如何添加自定义菜单项	310
9.4.1 添加和修改菜单项	310
9.4.2 汉化菜单项	313
9.5 为何数据图经常出现断图	314
9.5.1 数据图断图根本原因分析	314
9.5.2 数据图断图外部原因分析	317
9.6 本章小结	320
附录 A 触发器支持函数列表	321
附录 B Zabbix 系统中的单位符号	325
附录 C Zabbix Agent 监控项目关键字列表	327
附录 D Zabbix 支持的宏变量列表	336
参考文献	344

图 1-1 常见开源监控系统比较

第 1 章 Zabbix 系统介绍

本章首先介绍什么是 Zabbix 系统，它的特点以及它与其他开源监控系统比较有什么优缺点等；之后，简单介绍一下 Zabbix 系统中各个组件、体系结构及各个组件之间的关系；接下来，介绍安装和部署一套 Zabbix 系统对软硬环境有哪些要求，并在此基础上详细介绍如何安装和部署一套 Zabbix 系统；最后，对 Zabbix 系统中一些常见的概念做了简单的介绍和说明，为介绍后续章节的内容做一些必要的铺垫。

1.1 什么是 Zabbix 监控系统

Zabbix（音 zæbix）系统是一种企业级开源分布式监控解决方案。它所具有的丰富的数据采集方法使它几乎可以采集和处理所有类型的监控数据；而它所具有的灵活的报警机制，使它可以实现智能、灵活的报警策略；其 Web 组件不但方便我们日常管理和维护，而且可以输出近乎完美的数据图、拓扑图和各种报告；而它所特有的网络自动发现和低级自动发现功能，不仅能大大提高我们的工作效率，减少人为出错可能，而且使 Zabbix 系统相比其他开源监控系统具有更高的“智能”。

除了 Web 前端组件以外，Zabbix 系统的其他组件均使用 C/C++ 语言编写，这使得 Zabbix 系统具有非常高的运行效率；而其分布式的架构设计，不仅可以使它支持非常庞大的网络的监控，而且由此可以轻易实现跨地区、跨平台的分布式监控解决方案；Zabbix 系统数据的集中存储不仅方便我们日常的配置和管理，而且使对监控数据的进一步挖掘和分析成为可能（笔者认为，后续章节中将要介绍的“数据聚合”、“通过计算方式采集监控数据”均是对监控数据的再挖掘和再分析）。

Zabbix 系统不仅可以用来监控 CPU 负载、内存、磁盘使用率等这类常规的项目，而且它还可以监控 Web 站点的状态，甚至可以监控 SLA 信息，从而为 IT 基础设施架构的建设和改造提供数据支撑。

从上面的叙述可以看出，Zabbix 系统不愧为一款优秀的开源监控系统。虽说 Zabbix 系统是支持多国语言的，但是坦率地说，Zabbix 系统的汉化效果并不是很好，而且其中文资料也比较缺乏，通过互联网所能搜索到的关于 Zabbix 系统的中文资料，基本上都是网友编写的零散资料。在成稿时，笔者尚未见到国内较系统地介绍 Zabbix 系统的中文资料。虽然如此，它仍然是一款非常优秀的开源监控系统。况且，Zabbix 系统本身是支持多语言的，因此对它做进一步的汉化和完善也是一件非常容易的事（后续章节将详细介绍如何进一步完善 Zabbix 系统的汉化效果）。

Zabbix 系统最初是由 Alexei Vladishev 于 2001 年开发的，目前由 Zabbix SIA 公司在积极开发和维护。虽然目前 Zabbix 系统是由商业公司在开发和维护，但它仍然是一款免费的开源软件，它的开发和发布遵循“通用公共许可证（GPL）”V2 版。这就意味着所有人都可以自由地获取和使用 Zabbix 系统。

1.2 Zabbix 监控系统的特点

Zabbix 作为一款优秀的企业级开源监控系统,它能提供多达 13 种之多的监控数据采集方法,可以采集和监控 IT 基础设施中我们所需要监控的任何数据。同时,它所提供的 Web 组件和数据集中存储的方法,使我们日常管理和维护 Zabbix 系统变得非常简单。而 Zabbix 系统所提供的“网络自动发现”和“低级自动发现”功能更是可以大大提高我们的工作效率。具体来说,相较于其他开源的监控系统,Zabbix 系统具有以下特点:

- ❑ **丰富的数据采集方法。**Zabbix 系统提供多达 13 种之多的监控数据采集方法,可以采集 IT 基础设施中想要采集的几乎任何一种监控数据。
- ❑ **灵活和强大的报警机制可以实现智能的报警策略。**Zabbix 系统不但可以按照用户定义的报警计划、接收人和报警媒介发送报警信息,而且还可以实现报警级别的升级、报警信息重发次数的指定等功能。
- ❑ **易于管理和维护。**在 Zabbix 系统中,对所有被监控资源的管理和维护都是通过操作 Web 图形化界面来完成的,而不需要修改复杂的配置文件。更重要的是,Zabbix 系统支持“模板”和“关联”操作,因此,对 Zabbix 系统的管理和维护就显得非常简单和方便。
- ❑ **支持的平台很广泛。**Zabbix 服务器和服务器代理端可以安装在包括 Linux、AIX、FreeBSD、OpenBSD、Solaris 等在内的绝大多数类 UNIX 操作系统上;而其被监控设备代理 (Agent) 端则可以安装在 Windows 和绝大多数类 UNIX 操作系统上。
- ❑ **近乎完美的图形输出。**通过 Zabbix 系统的 Web 组件,不但可以实时查看到监控项目近乎完美的数据图,而且还可以实时查看网络拓扑图和各种图表。
- ❑ **智能的“网络自动发现”和“低级自动发现”功能。**使用 Zabbix 系统的“网络自动发现”和“低级自动发现”功能,不仅能大大提高我们的工作效率,减少人为出错可能,而且其使 Zabbix 系统相比其他开源的监控系统具有更高的“智能”。
- ❑ **丰富的 API 接口。**Zabbix 系统所提供丰富的 API 接口,不仅使我们对其进行二次开发成为可能,而且也方便我们将其与第三方软件进行整合。
- ❑ **高效性和支持分布式部署。**Zabbix 系统主要组件均使用 C/C++ 语言编写,这使得 Zabbix 系统具有非常高的运行效率。在一台非常普通的 PC 服务器上,Zabbix 系统每 ns 可以处理大约 15 000 个新数据。而其分布式的架构设计,不仅可以使它支持非常庞大的网络的监控,而且由此可以轻易实现跨地区、跨平台的分布式监控解决方案。

1.3 常见开源监控系统的比较

如今,监控系统种类繁多,既包括各种商业的监控软件,也包括越来越丰富的开源监控软件。这些监控系统不但架构各异,而且它们的功能也相差很大。表 1-1 列出了几款常见的开源监控系统,尝试比较一下它们各自的优缺点,以供读者参考。

表 1-1 常见开源监控系统比较

系统名称	Cacti	Nagios	Zabbix
运行平台	Linux、FreeBSD、Windows	服务器结点运行在Linux平台上。代理结点可以运行在Linux或Windows平台下	服务端运行在绝大多数类UNIX操作系统上，被监控设备代理端可以运行于Windows和绝大多数类UNIX操作系统上
软件功能	侧重于对网络流量、系统性能数据的采集和监控。使用插件可以对系统状态数据进行采集和监控	侧重于系统、网络状态的监控。虽可以对系统性能数据进行监控，但是其对性能数据的展示比较粗糙	能够对网络和系统状态、性能数据进行采集和监控。具有网络和监控项目低级自动发现功能。能够实施Web监控并具有主机资产管理功能
数据采集方法	主要依据SNMPGET抓取数据。也可以通过脚本抓取数据	可以通过SNMP协议、插件或用户自行编写的数据采集脚本进行数据采集	既可以支持SNMP协议抓取，也支持SNMP协议采集监控数据，还可以支持TCP或ICMP状态检测。同时，还支持通过IPMI、JMX、SSH、Zabbix代理和用户自定义脚本的方式采集数据
配置方式	通过Web界面配置，配置信息存放在MySQL数据库中	基于文本文件进行配置，配置比较复杂。商业版本可以支持通过Web界面配置，但是需要另外付费	通过Web界面配置，配置信息存储在MySQL等数据库中
开发语言	PHP（spine使用C/C++编写）	C/C++（Web前台使用Perl开发）。Windows下的代理端使用VC++开发	核心组件使用C/C++语言开发，Web前台使用PHP语言开发
数据存储方式	配置信息存放在MySQL数据库中。采集的监控数据以文件形式存放在磁盘上	采集的监控数据保存在文本文件中。通过第三方中间件可以将采集的监控数据存放在MySQL数据库中	配置信息和采集的监控数据均存放在MySQL等数据库中
系统特点	配置简单，界面友好。绘制的数据图漂亮、美观。侧重于网络流量和系统性能数据的抓取与绘制。但是不太适合实时性要求很高的状态监控	Nagios属于企业级开源监控软件，侧重于对系统或网络的状态进行监控。但是，其部分特性，如波动、新鲜度等在某些场景下比较适用	企业级的分布式开源监控系统。易于管理和配置，能生成比较漂亮的数据图。其自动发现功能可大大减少日常管理的工作量。众多的监控数据采集方式和其所提供的API接口可以为用户提供灵活的数据采集方式。分布式系统架构可以支持监控更多的被监控设备

1.4 Zabbix 系统组件及其体系结构

与其他许多监控系统类似，完整的 Zabbix 系统也是由多个系统组件组成的，包括 Zabbix 服务器(Zabbix Server)、Zabbix 服务器代理(Zabbix Proxy)、Web 前端组件(Zabbix Web Frontend)、

被监控设备代理 (Zabbix Agent) 以及数据库存储组件等。当然, 所有这些组件并不是在每个实际监控系统中都需要安装和部署, 可以根据实际应用场景和环境的需要, 选择只安装某些必需组件, 而非必需的组件则可以不安装。

需要说明的是, 因为英文中 Agent 和 Proxy 单词翻译成中文都有“代理”的意思。所以, 本书在接下来的章节中, 在不产生混淆的情况下, 将以“代理”分别指代服务器代理 (Proxy) 或被监控设备代理 (Agent)。读者需自己根据具体的语境判断所述的“代理”是指服务器代理 (Proxy) 还是指被监控设备代理 (Agent)。而在可能产生混淆的情况下, 将以 Proxy 表示服务器代理, 而 Agent 表示被监控设备代理。

1.4.1 Zabbix 系统组件介绍

Zabbix 系统中各个组件的功能及其作用叙述如下。

1. Zabbix 服务器 (Zabbix Server)

Zabbix 服务器是 Zabbix 系统的核心组件, 它接收被监控设备代理和服务器代理收集的监控数据和状态信息; 负责所有监控设备和监控项目配置、数据的存储与报表的生成和展示、报警的发送等。Zabbix 服务器组件是 Zabbix 系统的必需组件。

2. Zabbix 服务器代理 (Zabbix Proxy)

正如该组件的名字所表示的那样, 为了减轻 Zabbix 服务器的压力, Zabbix 服务器代理代替 Zabbix 服务器收集和接收被监控项目的监控数据, 并将数据发送到 Zabbix 服务器上, 由 Zabbix 服务器处理或报警。Zabbix 服务器代理并不是 Zabbix 的必需组件, 但是, 有了 Zabbix 服务器代理, 可以减轻 Zabbix 服务器的压力, 提高 Zabbix 系统的监控性能。

3. Web 前端组件 (Zabbix Web Frontend)

Web 前端组件为用户提供了配置监控信息和查看监控数据的 Web 接口。当 Zabbix 系统安装和部署完成之后, 用户的其他管理和维护操作都通过 Web 前端组件所提供的用户图形界面来完成。同时, 通过 Web 组件, 用户也可以随时随地查看被监控设备或被监控项目的监控数据和数据图。

在中小型的监控环境中, Web 前端组件一般被安装在与 Zabbix 服务器是同一台物理服务器上。当然, 这不是必需的, 换句话说, 实际上 Zabbix 服务器和 Web 前端组件可以分别部署在不同的服务器上。但是, 如果 Zabbix 系统数据库使用的是 SQLite, 则需要安装在同一台服务器上。

4. 被监控设备代理 (Zabbix Agent)

被监控设备代理组件是运行在被监控设备上的一个监控组件, 用以收集被监控设备上的各种监控数据, 并将这些监控数据发送给 Zabbix 服务器。被监控设备代理不是一个必需的组件。实际上, Zabbix 系统支持众多的监控数据采集方法, 通过被监控设备代理采集监控数据只是其中一种方法。

5. 数据存储系统

前面介绍过, 在 Zabbix 系统中, 所有被监控主机和被监控项目的配置信息以及系统所采集的所有监控数据都存储在数据库中, 所以, 数据库是 Zabbix 系统不可或缺的组成部分。Zabbix 系统可以支持包括 MySQL 数据库在内的多种关系数据库, 它们是 MySQL、Oracle、SQLite 及

Postgre SQL 等。

1.4.2 Zabbix 系统各组件之间的关系

Zabbix 服务器组件是 Zabbix 系统的核心组件，它负责监控数据的采集（当不使用服务器代理采集监控数据时）、收集（当使用服务器代理采集监控数据时）、分析处理与存储，并在必要的时候发送报警信息。

服务器代理接收来自 Zabbix 服务器的监控配置信息，并依据监控配置信息实施具体的监控和数据采集，并将所采集的数据依照一定的频率发送给 Zabbix 服务器，由 Zabbix 服务器分析处理和保存这些数据。

Web 前端组件接收用户指令，配置监控设备和监控项目信息；同时，依照用户的指令展示监控数据或图表。

被监控设备代理则主要负责本主机上监控数据的采集，并将所采集的监控数据定期发送给 Zabbix 服务器或服务器代理。

而数据库则提供数据的存储与查询服务，包括所配置的监控配置信息数据和所采集的监控数据等。

综上所述，Zabbix 系统中各组件的关系如图 1-1 所示。

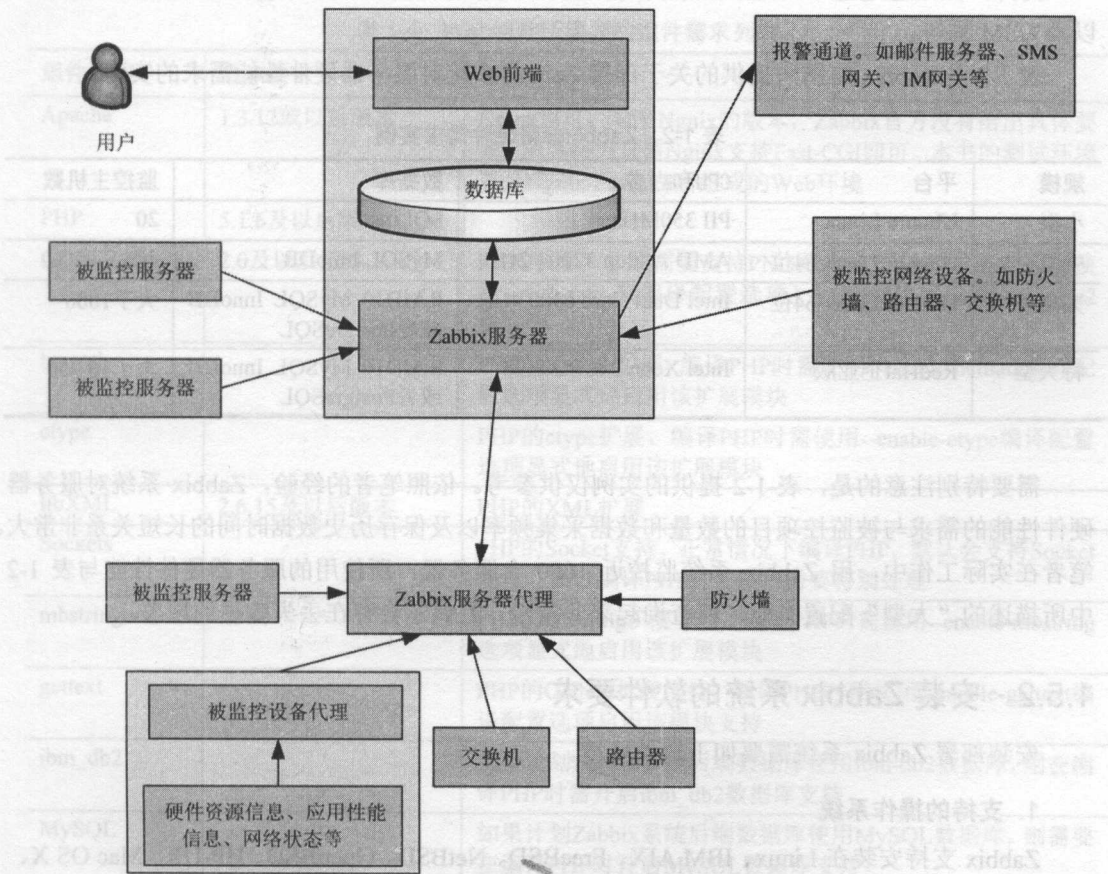


图 1-1 Zabbix 系统各组件关系图

1.5 部署 Zabbix 系统的软硬件需求

虽说如今的物理服务器基本上都能满足安装 Zabbix 系统对硬件的要求，但是，如果在虚拟机上安装 Zabbix 系统可能就不一样了。所以，在具体实施安装 Zabbix 系统之前，有必要先了解安装部署 Zabbix 系统都有哪些软硬件要求。

1.5.1 安装 Zabbix 系统的硬件需求

据 Zabbix 官网介绍，Zabbix 系统对服务器的硬件性能要求不高。一般有 128MB 的物理内存和 256MB 的磁盘空间就可以安装部署 Zabbix 系统了。很显然，Zabbix 系统对服务器硬件性能需求的高低，很大程度上是依赖于被监控设备和监控项目的数量及监控参数的设置而定的。例如，被监控设备和被监控项目数量都比较多，而数据采集的频率又很高，同时，所要求保留的历史记录时间也比较长，那么此时 Zabbix 系统对服务器的硬件性能要求也就相应地大幅提高。同时，Zabbix 系统所使用的数据库类型不同，例如 MySQL 或者 Oracle 等，以及所使用的存储引擎不同，对服务器硬件性能的要求也是不一样的。

另外，如果想通过 GSM 猫来发送报警短信，则服务器需要有串口或 USB 到串口转换接口以及 GSM 猫等。

表 1-2 是 Zabbix 官网所提供的关于部署 Zabbix 系统对服务器硬件性能需求的实例。

表 1-2 Zabbix部署硬件需求实例

规模	平台	CPU和内存	数据库	监控主机数
小型	Ubuntu Linux	PII 350MHz	SQLite	20
中型	Ubuntu Linux 64位	AMD Athlon 3200+ 2GB	MySQL InnoDB	500
大型	Ubuntu Linux 64位	Intel Dual Core 6400 4GB	RAID10 MySQL InnoDB 或者PostgreSQL	大于1000
特大型	RedHat企业版	Intel Xeon 2xCPU 8GB	RAID10 MySQL InnoDB 或者PostgreSQL	大于10 000

需要特别注意的是，表 1-2 提供的实例仅供参考。依照笔者的经验，Zabbix 系统对服务器硬件性能的需求与被监控项目的数量和数据采集频率以及保存历史数据时间的长短关系非常大。笔者在实际工作中，用 Zabbix 系统监控近 1000 台服务器，所使用的服务器硬件性能与表 1-2 中所描述的“大型”配置类似，但查询起来非常慢，且偶尔会存在丢失数据的现象。

1.5.2 安装 Zabbix 系统的软件要求

安装部署 Zabbix 系统需要如下软件环境。

1. 支持的操作系统

Zabbix 支持安装在 Linux、IBM AIX、FreeBSD、NetBSD、OpenBSD、HP-UX、Mac OS X、Solaris 等类 UNIX 操作系统上。Windows 2000、2003、XP、Vista、Server 2008、7、8 以及 Server 2012 等则只支持安装 Zabbix Agent 组件。

2. 支持的数据库

Zabbix 系统可以支持与表 1-3 中所示的关系型数据库中的任何一种协同工作。

表 1-3 Zabbix支持的数据库列表

数据库名称	最低版本要求	备注
MySQL	5.0.3 及以后版本	
Oracle	10g及以后版本	
PostgreSQL	8.1及以后版本	虽然Zabbix系统支持PostgreSQL，但是建议至少使用8.3及以上的版本
SQLite	3.3.5及以后版本	
IBM DB2	9.7及以后版本	

3. Web 前端组件对软件的需求

Zabbix 系统的 Web 前端组件是使用 PHP 语言开的，且它是 Zabbix 系统的一个必需组件，没有它基本上无法添加被监控设备或监控项目。因此，安装 Zabbix 系统 Web 前端组件所需要的软件环境基本上也就是运行 PHP 应用系统所需的环境，其具体所需的组件及对组件版本的要求如表 1-4 所示。

表 1-4 Web组件所需要的组件需求列表

组件名称	版本要求	备注
Apache	1.3.12或以后版本	Nginx也可。对于Ngnix的版本，Zabbix官方没有给出具体要求，笔者认为只需Nginx支持Fast-CGI即可。本书的测试环境就是Nginx-FastCGI所组成的Web环境
PHP	5.1.6及以后版本	
GD库支持	2.0及以后版本	PHP的GD库扩展需要支持PNG图片格式（编译PHP时需要使用--with-png-dir编译配置选项）、JPEG格式以及FreeType2格式等
bcmath		PHP的bcmath扩展，编译PHP时需使用--enable-bcmath编译配置选项显式地启用该扩展模块
ctype		PHP的ctype扩展，编译PHP时需使用--enable-ctype编译配置选项显式地启用该扩展模块
libXML	2.6.15及以后版本	PHP的XML扩展
Sockets		PHP的Socket支持，正常情况下编译PHP，默认会支持Socket功能，所以编译PHP时一般不需要特别处理
mbstring		PHP的mbstring扩展支持。编译PHP时需使用--enable-mbstring选项显式地启用该扩展模块
gettext		PHP的Gettext扩展支持。编译PHP时需使用--enable-gettext编译配置选项启用该模块支持
ibm_db2		如果计划Zabbix系统后端数据库使用ibm-db2数据库，则在编译PHP时需开启ibm_db2数据库支持
MySQL		如果计划Zabbix系统后端数据库使用MySQL数据库，则需要在编译PHP时开启MySQL数据库支持
oci8		如果计划Zabbix系统后端数据库使用Oracle数据库，则需要在编译PHP时开启Oracle数据库支持

续表

组件名称	版本要求	备注
pgsql		如果计划Zabbix系统后端数据库使用PostgreSQL数据库，则需要在编译PHP时开启PostgreSQL数据库支持
Sqlite3		如果计划Zabbix系统后端数据库使用SQLite数据库，则需要在编译PHP时开启SQLite数据库支持

当然，所安装的 Zabbix 系统的版本不同，对软件环境的要求，特别是对软件版本的要求也稍有不同。表 1-4 中所列出的是 Zabbix 2.0.0 版本对其他软件及版本的需求，Zabbix 的其他版本对软件环境的要求请参阅 Zabbix 官方手册。

4. 对浏览器的要求

要顺利地打开和浏览 Zabbix 系统的 Web 前端页面，则要求浏览器开启 Cookie 和 JavaScript 脚本支持。最新的谷歌、火狐、IE 等浏览器均支持 Zabbix 系统的 Web 端页面。

5. 其他软件需求

安装部署 Zabbix 系统除了需要上述组件外，还可能需要如表 1-5 所示的组件。为什么说“可能”还需要表 1-5 所列的组件呢？这是因为，表 1-5 所列的这些组件将依据 Zabbix 系统需要使用哪些监控数据采集方法而决定是否需要安装。如果 Zabbix 系统计划不使用对应的监控数据采集方法，则对应的组件就可以不安装。至于表 1-5 中所列的监控数据采集方法，读者暂时不理解也没有关系，本书的下一章将会专门介绍 Zabbix 系统中的监控数据采集方法，届时自然就会明白它们的含义和作用。

表 1-5 Zabbix系统对其他组件的要求列表

组件名称	备注
OpenIPMI	当需要使用IPMI监控数据采集方法时，则需要安装此组件，否则无须安装
libssh2	当需要使用SSH方法采集监控数据时，则需要安装此组件
Fping	Zabbix系统需要使用Fping组件来判断主机是否存活，故该组件一般要安装
libcurl	当你的Zabbix系统中有或者打算配置Web监控项目时，则需要安装该组件。Zabbix服务器使用Curl方法来请求所监控的Web页面
libksemel	如果打算使用Jabber方式报警，则需要安装此组件
net-snmp	如果打算使用SNMP协议采集监控数据，则需要安装此组件。对网络设备，例如路由器、交换机、防火墙等，通过SNMP协议采集监控数据可能是监控系统采集监控数据的唯一方法，因此该组件一般都需要安装

1.5.3 关于 Zabbix 系统数据库大小的计算

如果不是大批量地添加或删除被监控主机或监控项目，一般来说，Zabbix 系统中配置信息所占用数据库的空间是不大的，而且也不会有太大的变化。因而 Zabbix 系统数据库数据量的大小主要与数据库所存储的监控数据的数据量有关，而数据库中所存储的监控数据的数据量的大小又与下面这些配置有关。

1. 平均每 ns 处理数据的条数

平均每 ns 处理数据的条数是指, Zabbix 服务器(同样也适用于服务器代理)平均每 ns 接收和处理的新监控数据的条数。例如, 如果 Zabbix 系统监控了 3000 个监控项目, 而监控数据的采集频率是 60ns 采集一次, 则 Zabbix 系统平均每 ns 所要处理的新监控数据的条数为: $3000/60=50$ 。

2. 保留历史数据时长设置

Zabbix 系统保留历史监控数据的时间长短是可以设置的, 通常配置为几个星期或几个月。而每条新的监控数据及其索引都会占用一定的数据库空间。例如, 如果平均每秒处理 50 条监控数据, 历史数据保留 30 天, 那么系统中历史监控数据就将约有 1.3 亿条($30 \times 24 \times 3600 \times 50 = 12\,960\,000$)。由此可见, Zabbix 系统的数据量是非常可观的。

而历史监控数据所占用数据库空间的大小又与数据库的存储引擎、所存储的数据类型(例如浮点型数据、整型数据及字符串型数据等)有很大的关系。通常情况下, Zabbix 系统数据库存储单条监控数据所占用的磁盘空间大概在 40 字节到几百字节之间。根据经验统计分析, Zabbix 系统数据库存储单条监控数据平均所占用的磁盘空间大约为 50 个字节左右。据此, 如果 Zabbix 系统数据库中存储了上述的约 1.3 亿条历史监控数据, 则数据库需要占用磁盘空间大小大约为 6.5GB ($12\,960\,000 \times 60 = 6.5\text{GB}$)。

3. 保留趋势数据时长设置

Zabbix 系统会在系统数据库的趋势表中保存每个监控项目在每个小时内所采集到的全部监控数据中的最大值、最小值、平均值以及数据条数(当然, 如果在配置监控项目时指定保留 0 天趋势数据, 则系统就不会在数据库中保存这些数据), 这些数据即为趋势数据。这些数据主要是在系统绘制监控项目的长期数据趋势图时使用。因为 Zabbix 系统所保留的趋势数据是按每小时为间隔保存的, 所以, 趋势图的精度也就是一小时。

虽说每条趋势数据所需的存储空间大小会因为所使用的数据库类型的不同而稍有差异, 但是从估计的角度来说, 每条趋势数据大约会占用 128 字节的数据库空间。所以, 如果以上述监控 3000 个监控项目为例, 那么, 保留一年趋势数据所需要的数据库空间大约为 3.4GB ($3000 \times 24 \times 365 \times 128 = 3.4\text{GB}$)。

4. 事件数据保留时长设置

每个 Zabbix 系统事件数据大约需要使用 130 字节的磁盘空间来存储。虽然很难估计一台 Zabbix 服务器到底一天会产生多少个事件, 但是如果按照每 ns 产生一个事件来计算, 同时事件数据保留 3 年, 则保存这些事件数据所需要的数据库空间大约为 12.3GB ($3 \times 365 \times 24 \times 3600 \times 130 = 12.3\text{GB}$)。

综上所述, 整个 Zabbix 系统所需要使用的数据库空间大小可按如下公式计算:

所需总的数据库空间=存储配置信息所需的数据库空间(约 10MB 左右或更小)+存储历史数据所需的数据库空间+存储趋势数据所需的数据库空间+存储事件数据所需的数据库空间。

1.6 独立服务器安装与部署

Zabbix 系统的独立服务器模式是相对于多级分布式模式(何为单级分布式模式和多级分布

式模式，我们在后续章节中将有详细的介绍)而言的，它是指只有一台逻辑服务器结点的 Zabbix 系统工作模式。Zabbix 系统的这种工作模式适用于监控大中小型网络环境（而多级分布式工作模式则适用于特大型网络监控）。接下来就让我们安装部署一套 Zabbix 监控系统，一睹 Zabbix 监控系统的庐山真面目吧。

Zabbix 官方网站 (<http://www.zabbix.com>) 上提供了多种格式的 Zabbix 系统安装包供用户下载，包括源码包、编译好的二进制安装包，甚至还有多种虚拟机文件供用户下载。在这里我们将介绍，在实际生产环境中使用得比较多的，且在几种方式中比较复杂的一种安装方式，即通过源码编译的方式，安装部署 Zabbix 系统。

1.6.1 安装前准备

如前文所述，Zabbix 系统需要一些其他软件包支持才能安装和部署，因此，在具体安装 Zabbix 系统之前应安装这些软件或下载下来。故此，在具体安装部署 Zabbix 系统之前，需要做好两部分的准备工作。首先，需要将不必通过源码编译方式安装的软件包通过 yum 命令安装到目标系统上，具体需要执行的命令为：

```
shell>yum install beecrypt beecrypt-devel curl curl-devel freetype freetype-devel \
fontconfig fontconfig-devel gettext gettext-devel ImageMagick ImageMagick-devel \
mingw32-iconv.noarch libmcrypt libmcrypt-devel libpng libpng-devel libxml2 \
libxml2-devel libxslt libxslt-devel mhash mhash-devel mcrypt zlib zlib-devel gd \
gd-devel libjpeg libjpeg-devel
```

接下来，需要提前将安装过程中所需要的软件包下载到目标服务器上。下载方法比较简单，只需在目标服务器上执行类似 wget <http://URL> 命令即可下载指定的软件包，其中 URL 为所需要下载软件包的源 URL 地址。我们需要下载的软件及其下载地址如表 1-6 所示。

表 1-6 软件包下载地址列表

包名称	下载地址
MySQL5.6.5-M8	http://downloads.mysql.com/archives/get/file/mysql-5.6.5-m8.tar.gz
cmake-2.8.8	http://www.cmake.org/files/v2.8/cmake-2.8.8.tar.gz
php-5.4.29	http://cn2.php.net/get/php-5.4.29.tar.gz/from/this/mirror
libiconv-1.14	http://ftp.gnu.org/gnu/libiconv/libiconv-1.14.tar.gz
jdk-6u38	http://download.oracle.com/otn/java/jdk/6u38-b05/jdk-6u38-linux-x64.bin
nginx-1.2.9	http://nginx.org/download/nginx-1.2.9.tar.gz
iksemel-1.4	http://iksemel.googlecode.com/files/iksemel-1.4.tar.gz
libssh2-1.4.3	http://www.libssh2.org/download/libssh2-1.4.3.tar.gz
fping-2.4b2_to	http://download.chinaunix.net/down.php?id=6038&ResourceID=3222&site=1
OpenIPMI-2.0.21	http://downloads.sourceforge.net/project/openipmi/OpenIPMI%202.0%20Library/OpenIPMI-2.0.21.tar.gz?r=http%3A%2F%2Fsourceforge.net%2Fprojects%2Fopenipmi%2F&ts=1401936534&use_mirror=superb-dca3
Zabbix- 2.0.0rc1	http://downloads.sourceforge.net/project/zabbix/ZABBIX%20Release%20Candidates/2.0.0rc1/zabbix-2.0.0rc1.tar.gz?r=http%3A%2F%2Fsourceforge.net%2Fprojects%2Fzabbix%2Ffiles%2FZABBIX%2520Release%2520Candidates%2F2.0.0rc1%2F&ts=1402024764&use_mirror=heanet

提示：表 1-6 中所列的软件下载地址在本书编写的时候笔者均逐一验证过是可用的，但笔者不能保证这些下载地址会一直可用。如果在阅读本书时，发现某些地址不可用，也可以用对应软件的其他版本来代替，只需满足安装 Zabbix 系统所需要的软件环境即可。

1.6.2 LNMP 环境安装

Zabbix 系统的 Web 组件可以运行在 LNMP 或 LAMP 环境中，而当前 LNMP 环境似乎比 LAMP 环境更流行，故在这里就选择 LNMP 环境作为 Zabbix 系统的 Web 组件的运行环境。以下是我们在 CentOS 5.9 64 位、1GB RAM 系统环境下安装 LNMP 环境的详细步骤。

1. 安装 MySQL 数据库

因为 MySQL 从 5.5 版本开始，不再使用开源软件通行使用的 `./configure` 脚本来配置编译选项，而改用 `cmake` 命令来配置编译选项，而且 `cmake` 软件包在默认情况下是没有安装在操作系统上的，所以这里我们首先需要安装 `cmake` 软件包。

① 安装 cmake 软件包

```
shell> tar -zxvf cmake-2.8.8.tar.gz           #解压软件包
shell> cd cmake-2.8.8
shell> ./bootstrap
shell> gmake
shell> gmake install
```

执行完上述步骤后，使用 `cmake --version` 命令检查 `cmake` 安装是否正常。如果安装正常，则能显示 `cmake` 的版本号。

② 安装 MySQL

```
shell> tar -zxvf mysql-5.6.5-m8.tar.gz
shell> cd mysql-5.6.5-m8
shell> cmake -DCMAKE_INSTALL_PREFIX=/opt/mysql \
-DMySQL_UNIX_ADDR=/tmp/mysql.sock -DDEFAULT_CHARSET=utf8 \
-DDEFAULT_COLLATION=utf8_general_ci \
-DWITH_EXTRA_CHARSETS:STRING=all \
-DWITH_INNODB_STORAGE_ENGINE=1 \
-DWITH_ARCHIVE_STORAGE_ENGINE=1 \
-DWITH_READLINE=1 -DENABLED_LOCAL_INFILE=1 \
-DMySQL_USER=mysql -DMySQL_TCP_PORT=3306
shell> make                                     #编译
shell> make install                             #安装
shell> useradd -s /sbin/nologin mysql           #添加 mysql 用户
shell> cp support-files/my-large.cnf /etc/my.cnf #复制 mysql 配置文件到/etc 目录下
shell> cd /usr/local/server/mysql
shell> mkdir -p /data/mysql/data               #创建 mysql 数据目录
shell> scripts/mysql_install_db --datadir=/data/mysql/data/ \
--defaults-file=/etc/my.cnf --user=mysql       #初始化数据库
shell> cp support-files/mysql.server /etc/init.d/ #复制启动脚本到对应目录
shell> chkconfig --add mysql.server            #设置开机自动启动
shell> service mysql.server start              #启动 mysql
shell> echo "PATH=$PATH:/opt/mysql/bin" >>/etc/profile
shell> echo "export PATH" >>/etc/profile
#把 mysql 路径添加到 PATH 环境变量中
```

2. 安装 PHP

在安装 PHP 之前, 需要安装 libiconv 软件包 (该软件包的作用是转换语言编码), 其他 PHP 所需要的软件包在前面都已经使用 yum 命令安装完成了, 所以无须重复安装。

① 安装 libiconv 软件包

```
shell> tar -zxvf libiconv-1.14.tar.gz           #解压软件包
shell> cd libiconv-1.14
shell> ./configure --prefix=/usr/local          #执行编译配置
shell> make && make install                     #编译并安装
shell> ldconfig                                  #执行 ldconfig 命令, 更新动态库缓存
```

② 安装 PHP

```
shell> tar -zxvf php-5.4.5.tar.gz               #解压软件包
shell> cd php-5.4.5
shell> ./configure --prefix=/opt/php--disable-debug --disable-ipv6 \
--disable-rpath --enable-bcmath --enable-exif \
--enable-gd-native-ttf --enable-mbregex --enable-mbstring=all --enable-pcntl \
--enable-safe-mode --enable-shmop --enable-soap --enable-sockets \
--enable-xml --with-config-file-path=/opt/php/etc \
--with-curl --with-curlwrappers --with-freetype-dir --with-gd --with-gettext \
--with-iconv-dir=/usr/local --with-jpeg-dir --with-ldap \
--with-ldap-sasl --with-libdir=lib --with-libxml-dir=/usr --with-mcrypt \
--with-mhash --with-openssl --with-pdo-mysql --with-pear --with-png-dir \
--with-xmllrpc --with-zlib-dir --with-mysqli=/opt/mysql/bin/mysql_config \
--with-mysql=/opt/mysql \
--enable-fastcgi --enable-fpm --enable-force-cgi-redirect
#执行编译配置
shell> make ZEND_EXTRA_LIBS='-liconv'          #编译
shell> make install                             #安装
shell> cp php.ini-production /opt/php/etc/php.ini
#拷贝 PHP 配置文件到指定目录下
shell> cp /opt/php/etc/php-fpm.conf.default /opt/php/etc/php-fpm.conf
#按原始 php-fpm 配置文件复制一份到指定目录下。
```

3. 安装 Nginx

```
shell> tar -zxvf nginx-1.2.9.tar.gz             #解压软件包
shell> cd nginx-1.2.9
shell> ./configure --prefix=/opt/nginx --user=nobody --group=nobody \#编译配置
--with-poll_module --with-http_ssl_module --with-http_sub_module \
--with-http_perl_module --with-mail --with-pcre
shell> make                                     #编译
shell> make install                             #安装
```

4. 启停服务

当安装好 PHP 和 Nginx 后, 可以通过下列命令来启停 PHP 的 FPM 服务和 Nginx 服务:

```
shell> /opt/php/sbin/php-fpm                   #启动 PHP 的 FPM 服务
shell> killall php-fpm                         #关闭 PHP 的 FPM 服务
shell> /opt/nginx/sbin/nginx                   #开启 Nginx 服务
shell> /opt/nginx/sbin/nginx -s stop           #关闭 Nginx 服务
```

当然, 在这里给出的启停 PHP-FPM 和 Nginx 服务的方法都是最原始的方法, 如果觉得这种方法不方便, 也可以在互联网上搜索对应的启停脚本。在互联网上关于启停这两个服务的脚本还是非常容易找到的, 限于篇幅的原因, 我们在这里就不提供它们的代码了。

提示：安装好 LNMP 环境后，按道理应该对该环境的各个组件进行优化。关于 MySQL 数据库的优化，将在本书的第 8 章详细介绍。而关于 Nginx 和 PHP 的优化，一方面，一般来说 Zabbix 系统属于一个组织的内部系统，因此 Zabbix 系统 Web 组件的访问量不大，所以它不是我们调优的重点；另一方面，如今关于 LNMP 环境优化的资料已经非常多了，读者如果有这方面的需要，不难找到相关资料。

1.6.3 Zabbix 系统安装

在安装 Zabbix 系统之前，除了需要从 Zabbix 官方网站下载安装源码包之外，还需要计划好需要安装哪些组件以及可能会采用哪些方法采集监控数据（关于监控数据的采集方法，将会在下一章做详细的介绍）。下面介绍的安装步骤中，jdk、iksemel、libssh2、OpenIPMI 等软件包，可根据计划可能采用的监控数据采集方法选择是否安装。SNMP 协议是绝大多数设备都支持的标准协议，特别是网络设备和存储设备，一般来说，用户无法在其上安装其他的第三方数据采集软件。所以，要采集这类设备的监控数据，特别是性能数据，通过 SNMP 协议来采集是最方便也是最可行的方法。因此，建议在安装 Zabbix 服务器端时，SNMP 协议支持均选择上。在 Linux 系统中，要让 Zabbix 服务器支持 SNMP 协议采集监控数据，需要安装 net-snmp 和 net-snmp-devel 软件包。这两个软件包可通过 rpm 命令或 yum 源的方式来安装，其安装方法比较简单，请读者在具体安装 Zabbix 服务器端之前先行安装好这两个软件包，在此笔者就不一一细述两个软件包的安装过程了。

如前所述，Zabbix 系统自身带有 4 个组件。一般来说，在同一台服务器上这 4 个组件无须都安装。例如，Zabbix 服务器组件（Zabbix Server）和服务器代理组件（Zabbix Proxy），在同一台服务器上只需安装一个即可。

1. 安装 JDK 软件包

如果计划需要 Zabbix 系统通过 JMX 协议采集监控数据，那么就需要安装 Java 环境，即需要安装 JDK 软件包。安装 JDK 软件包比较简单，但是需要提请注意的是，操作系统的位数要和所安装的 JDK 软件包的位数相一致。以下是安装 JDK-6u38 软件包的简单步骤：

```
shell> chmod +x jdk-6u38-linux-x64.bin           #设置软件包可执行权限
shell> ./jdk-6u38-linux-x64.bin                   #执行软件包，让其自解压
shell> mv jdk1.6.0_38 /usr/java                   #将解压好的目录移到正式目录下
shell> echo "Java_HOME=/usr/java" >>/etc/profile  #设置 Java_HOME 环境变量
shell> echo "PATH=\$PATH:/usr/java/bin" >>/etc/profile
shell> echo "export Java_HOME" >>/etc/profile
shell> echo "export PATH " >>/etc/profile          #设置 PATH 环境变量
shell> source /etc/profile                        #使环境变量生效
```

2. 安装 iksemel 软件包

iksemel 软件包是用于使 Zabbix 系统支持通过 Jabber 方式发送报警信息的软件包，如果不打算通过 Jabber 方式来发送报警信息，那么这个软件包可以不用安装。具体何为 Jabber 方式，我们将在后续章节中做详细介绍。通常，很少通过 Jabber 方式发送报警信息，所以这个软件包一般可以不安装。

```
shell> tar -zxvf iksemel-1.4.tar.gz               #解压软件包
shell> cd iksemel-1.4
shell> ./configure --prefix=/usr/local/iksemel    #编译配置
```



```
shell> make && make install #编译并安装
```

3. 安装 libssh2 软件包

libssh2 是使 Zabbix 系统支持通过 SSH 协议采集监控数据所必需的软件包，如果不打算通过这种方式采集监控数据，那么这个软件包就可以不安装。但是，通过 SSH 方式运行用户自己编写的脚本来采集用户个性化的监控数据不失为一种比较好的方法，而且非常灵活。

```
shell> tar -zxvf libssh2-1.4.3.tar.gz #解压软件包
shell> cd libssh2-1.4.3
shell> ./configure #编译配置
shell> make && make install #编译并安装
```

4. 安装 OpenIPMI 软件包

OpenIPMI 软件包是让 Zabbix 系统支持通过 IPMI 方法采集监控数据所必需安装的软件包，如果不打算通过 IPMI 方式采集监控数据，那么这个软件包可以不安装。具体何为 IPMI 方法，将在后续章节中详细介绍。

```
shell> tar -zxvf OpenIPMI-2.0.21.tar.gz #解压软件包
shell> cd OpenIPMI-2.0.21
shell> ./configure #编译配置
shell> make && make install #编译并安装
```

5. 几个重要的 Zabbix 编译选项

与绝大多数数据开源软件一样，通过源码包方式编译安装 Zabbix 系统，首先也需要通过执行 configure 脚本程序来收集编译选项、系统软件环境并生成 Makefile 文件。然后，通过 make 和 make install 命令来编译并安装软件。在具体安装 Zabbix 系统之前，简单介绍 Zabbix 系统所特有的、对安装后的 Zabbix 系统有重要作用的几个编译配置选项。

- ❑ **--prefix=DIR**。指定 Zabbix 最终的安装路径。
- ❑ **--enable-static**。如果使用该编译选项，则表示使用静态方法编译 Zabbix 系统，否则为动态编译。静态编译后的二进制程序执行效率高，但是其体积也要大很多。
- ❑ **--enable-server**。如果使用该选项，则表示开启编译服务器组件开关。如果使用了这个选项，则在编译安装完成的 Zabbix 系统中会包含 Zabbix 服务器组件。
- ❑ **--enable-proxy**。如果使用该选项，则表示开启编译服务器代理组件开关。如果使用了这个选项，则在编译安装完成的 Zabbix 系统中会包含 Zabbix 服务器代理组件。
- ❑ **--enable-agent**。如果使用该选项，则表示开启编译被监控设备代理组件开关。如果使用了这个选项，则在编译安装完成的 Zabbix 系统中会包含 Zabbix 系统的被监控设备代理组件。
- ❑ **--enable-java**。如果使用该选项，则表示开启编译 Zabbix Java 网关组件开关。如果使用了这个选项，则在编译安装完成的 Zabbix 系统中会包含 Zabbix Java 网关应用程序。如果计划让 Zabbix 系统支持通过 JMX 协议采集监控数据，则需要显式地使用这个开关。
- ❑ **--with-mysql**。当 Zabbix 系统后端数据库使用 MySQL 数据库时，则用这个选项指定 MySQL 客户端动态库或 mysql_config 程序的路径，以便使编译后的 Zabbix 系统支持连接 MySQL 数据库。

- **-with-jabber**。该选项用于指定 iksemel 软件包所安装的路径。如果打算让 Zabbix 系统支持通过 Jabber 方式发送报警信息,则需要使用这个选项指定 iksemel 软件所安装的路径,如我们上面所安装的/usr/local/iksemel。

6. 准备数据库

很显然,要保证 Zabbix 服务器端组件能够正常连接并查询数据库,首先需要创建并初始化所需要的数据库。Zabbix 系统的数据库初始化文件在源码包的 database/mysql 目录下(以 MySQL 数据库为例)。以下是创建和初始化 Zabbix 数据库的简单操作步骤:

```
shell> /opt/mysql/bin/mysql -uroot -p           #连接到 MySQL 数据库
mysql> create database zabbix default character set=utf8; #创建 zabbix 数据库
mysql> grant all on zabbix.* to '<username>'@'<hostname>' identified by \
'<password>';
#创建 zabbix 访问数据库所需要的账号
mysql> quit;                                     #断开 MySQL 连接
shell> tar -zxvf zabbix-2.0.0rc1.tar.gz         #解压 Zabbix 源码包
shell> cd zabbix-2.0.0rc1/database/mysql
shell> mysql -u<username> -pzabbix <password> <schema.sql>
#导入 schema.sql 文件
shell> mysql -u<username> -p<password> zabbix <images.sql>
#导入 images.sql 文件
shell> mysql -u<username> -p<password> zabbix <data.sql> #导入 data.sql 文件
```

注意: 1. 数据库的导入是有先后顺序的,顺序错误将无法导入。

2. 如果所要导入的数据库只供服务器代理组件(Proxy)使用,则只需导入 schema.sql 文件,另外两个文件无须导入。

3. Zabbix 系统数据是采用 UTF8 编码的,所以在创建数据库时要设置数据库的默认编码为 UTF8,否则可能在 Web 前端或报警时出现乱码。

4. 以上介绍的是 MySQL 数据库下的数据导入方式,如果采用其他类型,则其数据导入方法请参阅 Zabbix 官网上的操作手册。

7. 安装 unixODBC 及数据库的驱动程序

Zabbix 系统提供一种被称之为“数据库监控”的监控数据采集方法。该数据采集方法的原理是,Zabbix 服务器进程直接通过 ODBC 接口技术,查询各种不同类型数据库里的数据,以采集需要的监控数据。因此,如果要让 Zabbix 系统能通过“数据库监控”方法采集监控数据,则需要安装 ODBC 中间件及相关的驱动程序。Zabbix 系统支持 unixODBC 和 iODBC 两种 ODBC 中间件。在这里我们以 unixODBC 为例,介绍如何安装配置 unixODBC 及数据库驱动程序。

unixODBC 软件包及 MySQL 数据库的 ODBC 驱动程序,在 CentOS 标准 yum 源里有。所以,在这里我们通过 yum 的方式来安装它们。

```
shell> yum -y install unixODBC unixODBC-devel
#安装 unixODBC 和 unixODBC-devel 两个软件包。
shell> yum install mysql-connector-odbc #安装 MySQL 数据库的 ODBC 驱动程序
```

上述步骤操作完成之后,安装脚本会在系统里创建/etc/odbcinst.ini 和/etc/odbc.ini 文件。请分别确认这两个文件里的内容是否是类似于下面的内容:

/etc/odbcinst.ini 文件内容:

```
[MySQL]
Description=ODBC for MySQL
Driver=/usr/lib/libmyodbc3_r.so
Setup=/usr/lib/libodbcmyS.so
Driver64=/usr/lib64/libmyodbc3_r.so
Setup64=/usr/lib64/libodbcmyS.so
FileUsage=1
```

/etc/odbc.ini 里的内容:

```
[mysql]
Description = Zabbix server MySQL database
Driver = MySQL
Server = 192.168.5.139
User = zabbix
Password = zabbix
Port = 3306
Socket = /tmp/mysql.sock
```

如果需要 Zabbix 系统能通过“数据库监控”的方法采集其他类型数据库里的数据,则需要相应地安装对应数据库的 ODBC 驱动程序。我们这里以 MS SQL Server 为例,介绍安装 ODBC 驱动程序的方法,其他类型数据库的 ODBC 驱动程序安装方法也类似。

首先下载驱动程序软件包,SQL Server、Access、Oracle、DB2 等数据库的 unixODBC 驱动程序软件包都可以从 http://www.easysoft.com/products/data_accessindex.html#odbc-drivers 页面下载。下载好驱动程序软件包后,按照下列步骤安装驱动程序:

```
shell> tar -xvf odbc-sqlserver-1.5.4-linux-x86-64-ul64.tar      #解压软件包
shell> cd odbc-sqlserver-1.5.4-linux-x86-64-ul64
shell> ./install                                              #执行安装脚本
```

执行上述命令后,根据提示安装 MS SQL Server 的驱动程序、配置证书、配置数据源等。如果安装正确,安装脚本会修改/etc/odbcinst.ini 文件,并添加如下的内容:

```
[Easysoft ODBC-SQL Server]
Driver=/usr/local/easysoft/sqlserver/lib/libessqlsrv.so
Setup=/usr/local/easysoft/sqlserver/lib/libessqlsrvS.so
Threading=0
FileUsage=1
DontDLClose=1
UsageCount=2
[Easysoft ODBC-SQL Server SSL]
Driver=/usr/local/easysoft/sqlserver/lib/libessqlsrv_ssl.so
Setup=/usr/local/easysoft/sqlserver/lib/libessqlsrvS.so
Threading=0
FileUsage=1
DontDLClose=1
UsageCount=2
```

同时,安装脚本会修改/etc/odbc.ini 文件,并添加如下的数据源信息:

```
[mssql]
Driver=Easysoft ODBC-SQL Server
Description=SQL Server DSN created during installation
Server=192.168.5.139
Port=1433
```



```
User=sqluser
Password=userpassword
Logging=0
QuotedId=Yes
AnsiNPF=Yes
Mars_Connection=No
```

安装配置完成之后, 可以使用 `isql -v mssql` 命令尝试用 ODBC 方式连接到指定的数据库。如果没有出错, 且可以正确执行 SQL 语句, 则说明 unixODBC 及 SQL Server 驱动程序安装配置都是正确的。

提示: 从 <http://www.easysoft.com> 网站下载的 ODBC 驱动程序均是只有 14 天试用期的试用版, 要长期使用, 则需要付费购买许可证。

8. 安装 Zabbix 软件包

经过上述的准备, 现在终于可以编译安装 Zabbix 服务器端及其他相关组件了。

① 创建 Zabbix 用户

默认情况下, Zabbix 系统的所有守护进程都是通过非特权用户运行的, 即使你使用特权用户 (root) 来启动 Zabbix 守护进程, Zabbix 进程在启动完成后也会自己切换成非特权用户。因此, 创建一个供 Zabbix 进程使用的非特权用户是必需的。

```
shell> groupadd zabbix #创建 zabbix 用户组
shell> useradd -g zabbix zabbix #创建 zabbix 用户
```

提示: 通过修改 Zabbix 组件配置文件中指定配置项的内容, 就可以使用特权用户 (root) 来运行 Zabbix 组件。关于需要修改哪个配置项的内容以及如何修改, 我们将在后续章节中再作详细介绍。

② 编译安装 Zabbix 组件

当某台服务器上同时运行 Zabbix 服务器和被监控设备代理服务器两个组件时, 建议使用两个不同的系统用户来分别运行这两个组件, 以免带来系统安全方面的风险。以下是编译安装 Zabbix 组件的具体步骤。因为在上述第 6 步已经将 Zabbix 源码包进行了解压, 所以在编译安装 Zabbix 组件时, 无须重复解压 Zabbix 源码包, 但是需要将当前工作目录切换到解压后的 Zabbix 源码包的根目录, 以下操作均是在该目录下进行的:

```
shell> ./configure --prefix=/opt/zabbix --enable-server --enable-agent \
--enable-java --with-mysql=/opt/mysql/bin/mysql_config \
--with-jabber=/usr/local/iksemel --with-net-snmp --with-libcurl \
--with-ssh2 --with-openipmi --with-unixodbc=/usr/bin/odbc_config #编译配置
shell> make #编译
shell> make install #安装
```

以上编译选项可以结合前面介绍过的配置选项的作用和功能与该台服务器在整个 Zabbix 系统中所担任的角色以及可能使用到的监控数据采集方法自由增减。

提示: 1. 当仅编译被监控设备代理组件 (agent) 时, 建议在编译配置时带上 `--enable-static` 配置选项以便使系统静态编译被监控设备代理组件。这样做有一个好处: 只需在一台机器上静态编译被监控设备代理组件, 其他机器只需将编译好的程序复制过去就可以使用了。

2. 如果在编译配置的时候使用了--enable-agent 配置选项, 那么命令行工具 zabbix_get 和 zabbix_sender 将会被编译安装。
3. 如果在编译配置时使用了--with-ucd-snmp 配置选项, 那么即使没有使用--with-net-snmp 配置选项, 编译后的 Zabbix 组件也将支持 SNMP 协议。所以, 如果想禁用 SNMP 协议采集监控数据, 则在编译配置时需要同时不使用--with-net-snmp 和--with-ucd-snmp 配置选项。

③ 修改配置文件

修改 Zabbix 服务器组件的配置文件 zabbix_server.conf, 这个文件在 Zabbix 安装目录(也就是上面在编译配置阶段使用--prefix 配置选项指定的路径)的 etc 目录下。这个配置文件中需要修改和确认的内容主要有以下几行:

```
# DBHost=localhost 行, 除掉该行最前面的注释符“#”变为 DBHost=localhost;
DBName=zabbix 行, 确认该行 DBName 配置项所指定的数据库名是否是 zabbix;
DBUser=root 行, 将该行 DBUser 配置项所指定的用户名修改为上面第 6 步所创建的 MySQL 用户名;
# DBPassword=行, 将该行最前面的注释符“#”除掉, 并将配置项 DBPassword 所指定的连接数据库密码修改为上面第 6 步所创建的 MySQL 数据库用户所对应的密码。
```

在小规模监控场景下, 例如只监控几台或十几台服务器和网络设备, zabbix_server.conf 配置文件里的其他配置项基本可以不做修改。但是, 如果要想让 Zabbix 系统发挥出最大的性能, 以使其能监控更多的网络设备或服务, 需要对 Zabbix 系统进行调优。至于如何针对 Zabbix 系统进行调优, 将在本书的后续章节中详细讨论。

如果编译安装了被监控设备代理组件, 则同样需要修改被监控设备代理组件的配置文件 zabbix_agentd.conf。这个文件也在 Zabbix 安装目录(即上面在编译配置阶段使用--prefix 选项指定的路径)的 etc 目录下。在这个配置文件中主要需要修改 Server 配置项的内容, 将其配置为 Zabbix 服务器端组件所在服务器的 IP 地址。

如果编译安装了 Zabbix 服务器代理(Proxy)组件, 则同样需要修改对应配置文件 zabbix_proxy.conf 的内容。在这个配置文件中主要需要修改服务器端组件所在服务器的 IP 地址(配置项为 Server)、服务器代理的名称(配置项为 Hostname)、数据库名(配置项为 DBName)、数据库用户名(配置项为 DBUser)和数据库密码(DBPassword)等。

④ 配置启动脚本

Zabbix 源码包里提供了大多数操作系统下启动 Zabbix 服务器组件或服务器代理组件的脚本。通过这些脚本不但可以很方便地控制这些组件的启停, 而且可以让这些组件能够随着操作系统一起启停。这个文件在 Zabbix 源码包的 misc/init.d 目录下。因为我们这里使用的是 CentOS 5.9 操作系统, 所以选用与 CentOS 最相近的 Fedora 版本的启动脚本。

```
shell> cp misc/init.d/fedora/core5/zabbix_server /etc/init.d/
shell> cp misc/init.d/fedora/core5/zabbix_agentd /etc/init.d/
```

修改这两个启动脚本文件中的 ZABBIX_BIN 变量值为 Zabbix 实际的安装路径, CONFIG_FILE 变量值为配置文件实际路径。然后执行下面命令将 Zabbix 服务添加为开机自动启动服务。

```
shell> chkconfig --add zabbix_server
shell> chkconfig --add zabbix_agentd
```

很是奇怪, Zabbix 源码包里提供了 Server 和 Agent 组件的启动脚本, 但是笔者在其源码包里却没有找到 Zabbix 服务器代理组件的启动脚本。不过没有关系, 其实只需将 zabbix_server 文件稍作修改, 将程序名修改为 Zabbix_Proxy, 便可将执行文件路径修改为 zabbix_proxy 文件

路径；配置文件路径修改为 `zabbix_proxy.conf` 文件路径等，即可将其作为 Zabbix 服务器代理组件的启动脚本来使用。

接下来，执行以下命令启动 Zabbix 服务器组件和被监控设备代理组件：

```
shell> service zabbix_server start          #启动 Zabbix 服务器组件
shell> service zabbix_agentd start         #启动被监控设备代理组件
```

9. 编译安装 fping 工具包

Zabbix 系统需要使用 `fping` 命令来判断被监控主机是否存活，故一般均需安装 `fping` 组件，以下是其安装步骤：

```
shell> tar -zxvf fping-2.4b2_to.tar.gz      #解压软件包
shell> cd fping-2.4b2_to
shell> ./configure                          #执行编译配置
shell> make && make install                 #编译并安装
```

1.6.4 部署 Web 前端组件

Zabbix 系统的 Web 前端组件是用 PHP 开发的，因而其与其他 PHP 应用系统类似，部署相对来说比较简单，而且 Zabbix 系统还提供一个部署其 Web 前端组件的安装向导。接下来，简单介绍如何安装和部署 Zabbix 系统的 Web 前端组件。

1. 配置虚拟主机

首先，需要在 Nginx 上配置一个虚拟主机，以用于运行 Zabbix 系统的 Web 前端组件。当然，如果用于运行 Zabbix 系统 Web 前端组件服务器上的 Nginx 只用运行 Zabbix 系统 Web 前端组件，则也可不设置虚拟主机，只需修改 Nginx 默认站点的站点路径。

编辑 Nginx 安装目录（也就是前面用 `--prefix` 选项所指定的路径）下的 `conf/nginx.conf` 文件，并在该文件中添加以下内容：

```
server {
    listen 80;
    server_name <HOSTNAME>;          #主机名
    access_log off;
    charset utf8;
    autoindex off;
    autoindex_exact_size off;
    autoindex_localtime on;

    location / {
        index index.html index.htm index.php;
        root /data/website/zabbix;    #站点路径
    }

    location ~ /\.php$ {
        include fastcgi_params;
        fastcgi_param SERVER_NAME $server_addr;
        fastcgi_pass 127.0.0.1:9000;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME $document_root/$fastcgi_script_name;
    }
}
```

重启 Nginx 以及 php-fpm:


```
shell> /opt/nginx/sbin/nginx -s reload #重新加载 nginx 配置文件
shell> killall php-fpm #关闭所有的 php-fpm 进程
shell> /opt/php/sbin/php-fpm #启动 php-fpm 进程
```

2. 复制 PHP 文件

Zabbix 系统 Web 前端组件是用 PHP 语言编写的，所以部署它就需要将该组件所对应的程序文件复制到站点的根目录下并做相应的修改。Zabbix 系统 Web 前端组件程序文件的存放目录为 Zabbix 源码包的 frontends/php 目录。

```
shell> mkdir -p /data/website/zabbix #创建站点目录
shell> cp -Rpfv frontends/php/* /data/website/zabbix/#复制站点程序文件到站点目录
shell> chown -R nobody:nobody /data/website/zabbix #修改站点目录权限
```

3. 修改 PHP 配置

完成上述步骤后，就可以尝试通过浏览器打开 Web 前端组件安装向导，进行安装部署。安装向导的打开地址是 http://hostname，这里的 hostname 就是前面配置虚拟主机时 server_name 配置项中所指定的 hostname。或者如果没有配置虚拟主机，则 hostname 是服务器的 IP 地址。

然而，因为我们所使用的 PHP 配置文件 php.ini 是直接复制 PHP 源码包中所提供的配置文件，这个配置文件里的某些配置项没有达到 Zabbix 系统所要求的最低配置，所以在用上述方法打开安装向导页面时，系统会显示如图 1-2 和图 1-3 所示的错误信息。

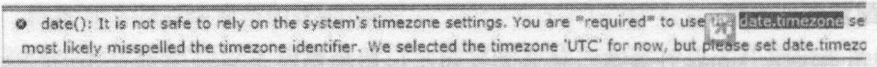


图 1-2 未配置时区错误信息

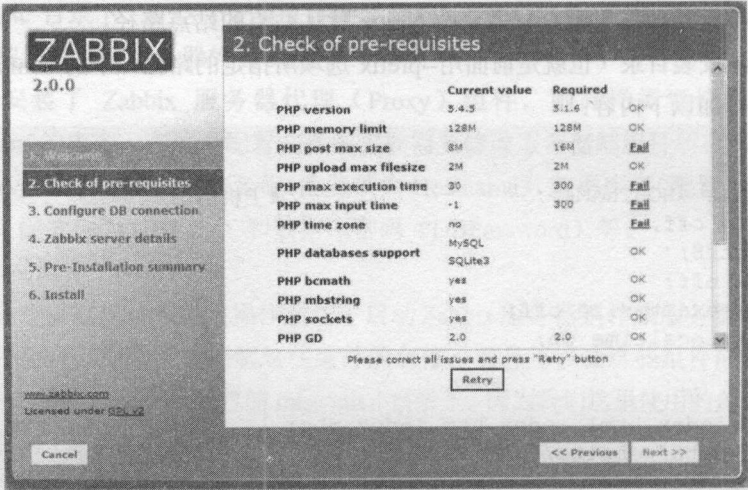


图 1-3 PHP 配置未达到安装最低要求信息

图 1-2 显示的错误信息是当未对 php.ini 文件中 date.timezone 配置项进行配置，而在程序中使用 date()函数时，PHP 所抛出的警告信息。图 1-3 显示的是当前 PHP 配置中，达到或未达到 Zabbix 系统所要求的最低配置的配置项列表信息。红色显示的配置项为未达到系统最低配置要求的配置项，必须修改这些配置项的配置并重启 php-fpm 服务，安装才能继续。修改 /opt/php/etc/php.ini 文件里的配置项，以使 PHP 的配置达到 Zabbix 系统所要求的最低配置，并重启 php-fpm 服务。表 1-7 列出了 Zabbix 系统对 PHP 配置的最低要求。

表 1-7 Zabbix系统对PHP配置的最低要求列表

配置项名称	最低要求配置	配置方法
PHP版本	5.3.0	
时区配置	不能为空，需要设定时区	php.ini文件中，修改date.timezone = Asia/Shanghai
内存限制	128MB	php.ini文件中，修改memory_limit = 128M
最大提交数据大小	16MB	php.ini文件中，修改post_max_size = 16M
最大上传文件大小	2MB	php.ini文件中，修改upload_max_filesize = 2M
最大执行时间	300秒	php.ini文件中，修改max_execution_time = 300
最大输入时间	300秒	php.ini文件中，修改max_input_time = 300
自动开启session	禁止	php.ini文件中，修改session.auto_start = 0

4. 安装 Web 前端组件

修改了 php.ini 文件后，使用下列命令重启 php-fpm 服务：

```
shell> killall php-fpm #关闭 php-fpm
shell> /opt/php/sbin/php-fpm #重新开启 php-fpm 服务
```

然后，单击浏览器上的刷新按钮，将出现如图 1-4 所示的安装向导。

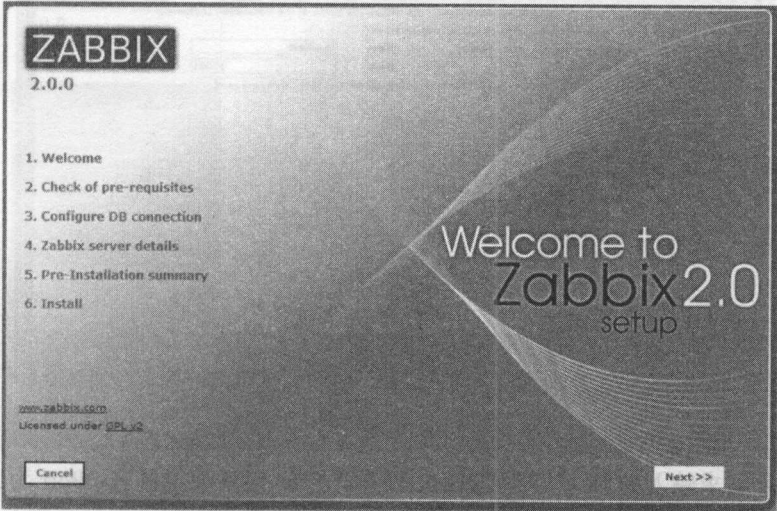


图 1-4 Zabbix 安装向导

单击 Next 按钮，系统将会再次显示类似图 1-3 所示的对安装环境的检查结果。在所有安装要求中，只要有一项没有达到最低安装要求，安装都无法继续进行。只有所有项目都达到了 Zabbix 系统所要求的最低安装要求后，才能进行下一步。当所有的要求项都达到了 Zabbix 系统所要求的最低安装要求时，单击 Next 按钮进行数据库参数的配置，如图 1-5 所示。

在配置数据库连接页面上选择数据库类型（这里选择 MySQL 数据库）、输入数据库的主机名或服务器 IP 地址、数据库端口（如果数据库端口是默认的 3306，则数据库端口项可以不填写）、数据库名称、连接数据库的用户名和密码等信息。输入完数据库连接参数以后，单击 Test Connection 按钮，测试系统使用所输入的数据库参数是否可以连接到数据库。

完成上述的数据库参数配置，并在测试连接正常的情况下，单击 Next 按钮进入 Zabbix 服务器端参数配置界面，如图 1-6 所示。

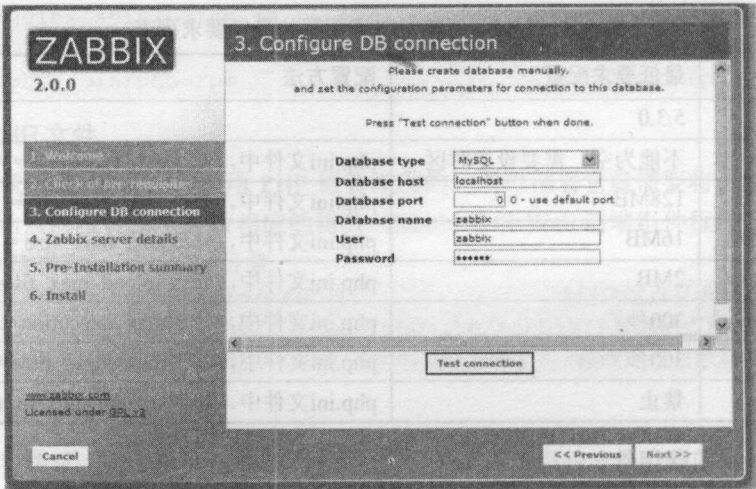


图 1-5 配置数据库参数

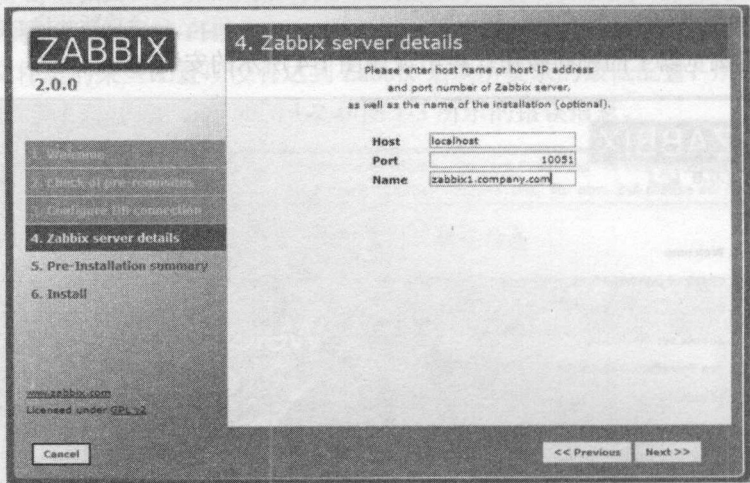


图 1-6 Zabbix 服务器参数配置

图 1-6 中所要求输入的 Host 即为 Zabbix 服务器的 IP 地址或主机名，Port 为 Zabbix 服务器端服务的端口号，默认为 10051；如果在 zabbix_server.conf 文件里对 Zabbix 服务器的端口号有所修改，则这里也需要做出相应的修改。Name 为 Zabbix 服务器的名字，可以不用填写。需要说明的是，这里所输入的 Zabbix 服务器名字不一定是 Zabbix 服务器端所在服务器的主机名，可以根据兴趣另外取一个新的名字，该名字主要是用于 Web 前端页面上显示使用的。输入完成上述三个参数后单击 Next 按钮，进入配置信息确认页面，如图 1-7 所示。

仔细检查参数确认页面里所显示的配置信息是否都正确，如果有不正确的参数，则需要返回前面相应页面进行修改，否则单击 Next 按钮，将配置信息写入到配置文件中，如图 1-8 所示。

如图 1-8 所示，如果配置信息写入配置文件正确，则单击 Finish 按钮，完成安装向导的工作。至此，Web 前端组件安装配置完成。如果配置信息写入配置文件出错，则需要确认出错的原因。一般配置信息写入配置文件出错的原因是由于 Web 站点所对应目录权限设置不正确，此时，需要先行将权限修改正确后再次启动安装向导并重新安装 Web 前端组件。

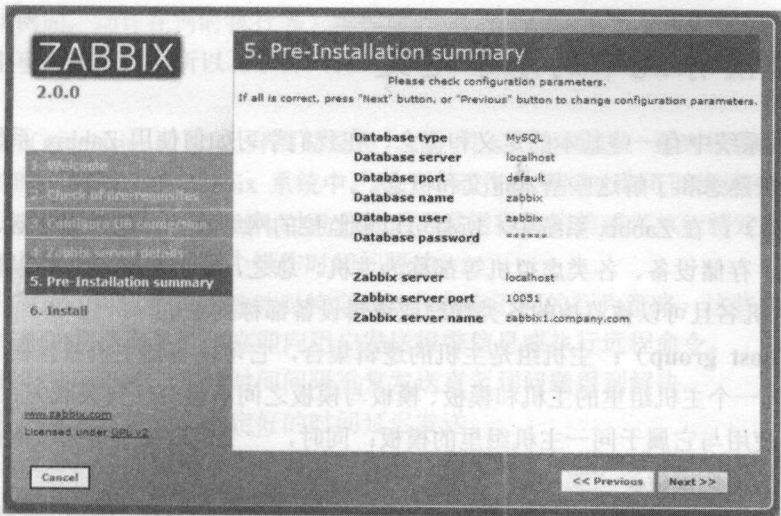


图 1-7 参数信息确认

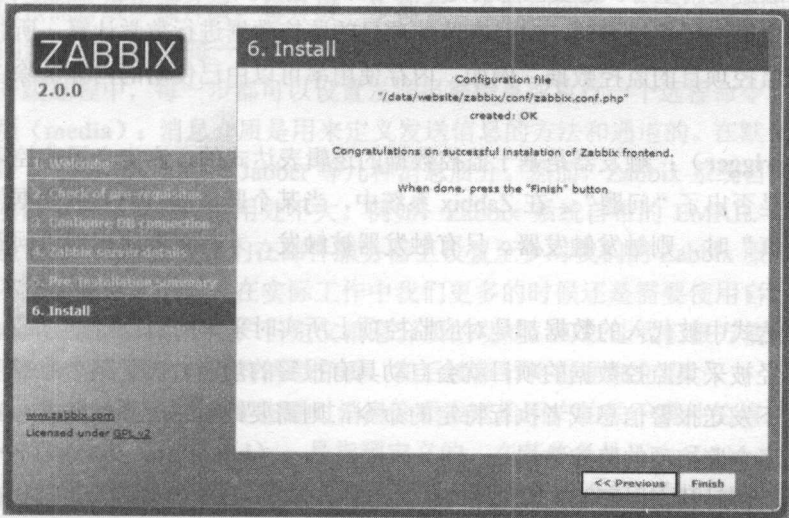


图 1-8 配置信息写入配置文件

完成安装向导的各个步骤的配置之后，安装向导将引导我们登录到系统中。Zabbix 系统默认给我们配置的账号为 Admin，密码为 zabbix。Zabbix 系统登录页面如图 1-9 所示。

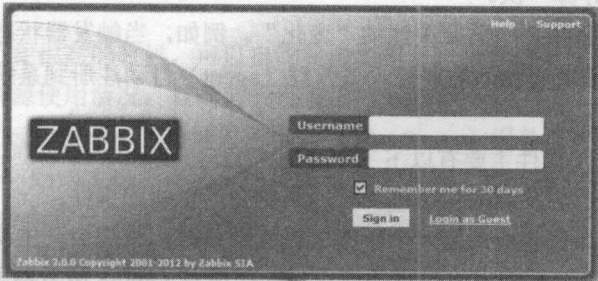


图 1-9 系统登录页面

1.7 Zabbix 系统中的基本定义

在 Zabbix 系统中有一些基本的定义和概念，在我们学习如何使用 Zabbix 系统之前，让我们花一点时间来熟悉和了解这些基本定义和概念。

主机 (host)：在 Zabbix 系统中，所有可以被监控的网络设备，包括服务器、交换机、路由器、防火墙、存储设备、各类虚拟机等都称为主机。总之，在监控网络中可以被独立配置网络 IP 地址或主机名且可以被监控的各类物理或逻辑设备都称为主机。

主机组 (host group)：主机组是主机的逻辑集合，它可以包括主机或模板。但是，需要注意的是，在同一个主机组里的主机和模板、模板与模板之间不得有任何关联关系。也就是说，一台主机不得使用与它属于同一主机组里的模板；同时，一个模板也不得关联或被关联到处于同一主机组里的其他模板。

监控项 (item)：监控项的概念很好理解，它就是需要监控的对象。例如，系统 CPU 负载、已使用内存、磁盘使用量等。需要注意的是，在 Zabbix 系统中，并不是所有监控项目的监控数据都是直接从主机上采集过来的。对已有监控项目的监控数据进行数学计算，所获得的数据可以作为另一个监控项目的监控数据。例如，内存使用率可以由已使用的内存量除以总的内存容量得出。

触发器 (trigger)：触发器是基于监控数据的逻辑表达式的，它定义了监控项的阈值，用以判断监控项是否出了“问题”。在 Zabbix 系统中，当某个监控项目的监控数据使某个触发器的表达式为“真”时，则触发触发器。只有触发器被触发，才有可能将出现“问题”的监控项目信息报警出去。

触发器表达式中被代入的数据都是对应监控项上所实时采集的监控数据。在 Zabbix 系统中，并不是所有已经被采集监控数据的项目就会自动具有报警的功能。对于某个监控项，如果要在某种情况下发送报警信息或者执行特定的命令，则需要针对它配置触发器。当然，一个触发器可以引用多个监控项的监控数据。

触发器具有两种状态，即 OK 状态和 PROBLEM 状态。OK 状态即为正常状态；而当触发器处于 PROBLEM 状态时，则表示对应触发器被触发了。触发器的状态会在每次 Zabbix 系统针对监控项采集到新的数据时重新计算。如果在触发器表达式中使用了 `nodata()`、`date()`、`dayofmonth()`、`dayofweek()`、`time()` 及 `now()` 等时间函数，则触发器的状态会每隔 30s 被 Zabbix 系统中的定时器重新计算一次。

事件 (event)：是指那些需要关注的“变化”。例如，当触发器状态改变，系统自动发现了新主机或监控项，或者有新的被监控设备代理 (agent) 自动注册到系统时，系统都会产生相应的事件。

在 Zabbix 系统中，事件主要有以下三个方面来源。

- **触发器。**每当触发器的状态改变时，系统会产生新的事件。
- **自动发现。**当自动检测到一个新的主机或服务时，系统会产生新的事件。
- **自动注册。**当有新的被监控设备代理自动注册到 Zabbix 服务器或服务器代理端时，系统会产生新的事件。

动作 (action)：是指对事件所预设的反应方法。动作由“行为”（例如发送报警信息等）

和“条件”（例如，动作在何时执行等）所构成。

动作是由事件驱动的，所以与事件一样，动作的触发源也是触发器、自动发现、自动注册等。

上升周期 (escalations)：上升周期是指，在一个动作中发送一系列报警信息或执行一系列远程命令的时间间隔。在 Zabbix 系统中，同一个动作在不同时段可以执行不同级别的行为。例如，在动作初始时刻发送报警邮件，过几分钟后再发送报警短信或者执行特定的远程命令等。而上升周期就是用于定义前后两个操作时间间隔的。

通过上升周期，可以很灵活地针对特定的动作制定不同的行为策略。这些策略可以包括：

- 在有新的问题产生时，立即向用户发送报警信息或执行远程命令。
- 报警信息可以按一定的时间间隔重复发送直至问题得到解决。
- 报警信息可以按预先设定好的时间延迟发送。
- 可以设定，当问题持续了一定的时间后，向更高一级用户或用户组发送报警信息或执行“更高一级”的远程命令。
- 可以设定，当问题解决时，系统自动给指定的用户发送恢复信息。

动作可以按照升级步骤升级，每升级一步都有一个时间间隔。而这个时间间隔可以为设置的默认时间间隔，也可以是每一步自己独立设置的时间间隔。动作的最小升级时间间隔是 60 秒。在动作升级过程中，每一步都可以设置发送报警信息或执行一个远程命令。

消息介质 (media)：消息介质是用来定义发送信息的方法和通道的。在默认安装的 Zabbix 系统中，自带了 EMAIL、SMS、Jabber 等几种消息介质。然而，Zabbix 系统自带的这几种消息介质在我们实际生产环境中可能用处不大。例如，Zabbix 系统自带的 EMAIL 消息介质是不支持发送邮件验证的，这就需要在邮件服务器上设置至少对我们的 Zabbix 服务器开放邮件中继，否则发不出去邮件。所以，在实际工作中我们更多的时候还是需要使用自己定义的消息介质。当然，Zabbix 系统提供了用户自定义消息介质的功能，可以让我们很方便地定义自己的消息介质。关于如何定义我们自己的消息介质，在后续章节将有详细介绍。

通知 (notification)：通知是指通过消息介质发送给用户的关于事件的信息。

远程命令 (remote command)：是指预定义的，在某些条件下由 Zabbix 系统在被监控主机上执行的命令。远程命令为智能处理某些监控事件提供了一个很好的机制。在以下这些场景下，使用远程命令机制，可能会带来非常明显的优势：

- 当某些应用（Web 服务器、中间件等）无法响应的时候，使用远程命令来重启它们。
- 当服务器无法响应时，使用 IPMI 的“reboot”命令来自动远程重启它们。
- 当磁盘空间快满的时候，可以自动地清理磁盘空间（例如删除老文件，清理/tmp 文件系统等）。
- 当宿主机负载很高的时候，可以自动将虚拟机从一台物理机上迁移到另一台物理机上。
- 根据系统资源使用情况，自动向云环境中增减结点。

需要注意的是，远程命令不支持通过服务器代理（Proxy）组件监控的主机。也就是说，如果需要使用过程命令，则该主机不得通过服务器代理来监控，而只能通过 Zabbix 服务器直接进行监控。另外，远程命令的长度最长不得超过 255 个字符。但是，远程命令支持执行多条命令，当有多条命令的时候，不同的命令分不同行书写，Zabbix 系统会逐条执行。同时，远程命令可以支持宏变量。

模板 (template)：可以应用到主机上的实体（包括监控项、触发器、数据图、图表、监

控分类、低级的自动发现规则)的集合。虽然在 Zabbix 系统里可以针对主机单独配置监控项、触发器、数据图等监控属性。但是,很显然,通过模板来管理和配置这些监控属性要方便得多,效率也要好得多,特别是在实际工作中,可能有大量功能和作用很相近的主机,通过模板来配置这些监控属性,然后分别应用到对应的主机上显然很方便,而且也便于管理。

当一个模板被应用到一个主机后,该模板的所有实体(包括监控项、触发器、数据图等)都会被添加到对应的主机上。不过,模板是需要直接应用到主机上的,而不能应用到主机组上。同时,模板和主机是多对多的关系,即一个模板可以应用到多个不同的主机上;而同一个主机又可以关联多个模板。所以,我们通常按服务的种类来创建模板,例如 Apache 模板、MySQL 模板、PostgreSQL 模板,等等。这样,当某个主机上运行某个服务的时候,只需将对应的模板分别应用到该主机即可。

另外,使用模板还有另外一个好处。当需要修改某个监控属性时,只需在对应的模板上做相应的修改即可,修改后的属性将会自动更改到应用了该模板的所有主机上。

监控分类(application): 监控分类是指监控项目的逻辑分组。也就是说,可以将具有相同或相近属性的监控项目划分在同一个分组里,这样有利于我们管理和查找,特别是当主机的监控项目很多的时候。

Web 场景(web scenario): Zabbix 系统可以检查 Web 站点的可用性。为了监控 Web 站点,需要定义 Web 场景。一个 Web 场景由一个或多个 HTTP 请求和“步骤”组成。“步骤”将由 Zabbix 服务器按预定的顺序执行。

通过 Web 场景,可以收集以下信息:

- 整个场景中所有步骤的平均下载速率
- 出错的步骤数
- 最后一次出错的信息

通过步骤,可以收集以下信息:

- 每 ns 下载的速率
- 响应时间
- 响应代码

Zabbix 系统可以检查指定的 HTML 页面中是否包含指定的字符串,这个功能一般用来检查 Web 站点是否可用。此外,Zabbix 系统还可以模拟登录以及模拟鼠标单击页面动作。

Zabbix 系统监控 Web 站点,既可以支持 HTTP 协议,也可以支持 HTTPS 协议请求指定的页面。同时,Zabbix 还可以很好地跟随页面跳转。在每次执行一个 Web 场景的检查过程中,Zabbix 系统能做到 cookie 保持。

注意: 如果要通过 HTTP 代理检查 Web 站点,那么需要在运行 Zabbix 服务器端的用户(一般用户名是 zabbix)的系统环境中添加 `http_proxy=http://proxy_ip:proxy_port` 环境变量。同样,如果要通过 HTTPS 代理检查 Web 站点,那么也需要在运行 Zabbix 服务器程序的用户(一般用户名是 zabbix)的系统环境中添加 `HTTPS_PROXY= http://proxy_ip:proxy_port` 环境变量。

结点(node): 在分布式的监控环境中,可以使用结点来部署有层次的监控系统。每个 Zabbix 结点都是一套完整的 Zabbix 服务器,并且它们负责各自领域主机的监控。在分布式环境中,Zabbix 系统可以支持多达数千个结点。使用结点具有以下好处:

- 使用结点可以构建一个多级的、适应于跨多地区的大型网络的监控系统。每个子结点仅与它自己的主结点通信。
- 子结点所管理的监控资源，如被监控主机、监控项目等，既可以通过其自身来管理和配置，也可以通过它的主结点来管理和配置，主结点仅仅保存它所有子结点所管理的监控资源的配置副本。
- 在本地监控数据的采集中，因为网络原因而导致的影响大大减小。即使主结点和子结点之间发生了通信故障，也不会影响子结点的使用。
- 通过结点可以大大增加 Zabbix 系统最多可监控主机的数量。
- 增减子结点不会对其他结点带来任何影响。

服务等级协议 (SLA)：在 Zabbix 系统的 IT 服务功能中，我们可能会接触到“服务等级协议 (Service-Level Agreement, SLA)”的概念，在此简单介绍一下。服务等级协议，它与 TCP/IP、HTTP、FTP 等网络通信协议不同，它是商业层面上的协议。从这个角度来说，称它为服务等级合同或许更贴切一些。因为它一般规定了网络服务供应商给用户提供的服务类型、服务质量、付款事宜，等等。在 Zabbix 系统中所涉及的服务等级协议没有这么复杂。在 Zabbix 系统中，所谓 SLA 是指对指定服务的可用性统计，以及对对应服务的可用性是否达到了最初设定的目标。在 Zabbix 系统中，判断某个服务是否可用的依据是对应触发器是否被触发，以及触发器被触发的持续时间。或许，你看了前面这些介绍对于什么是 SLA，特别是对于 Zabbix 系统中如何使用 SLA 还是一头雾水。没有关系，等到具体介绍配置 IT 服务时，你再回过头来看看这里介绍的内容，应该就能弄明白什么是 SLA 了。

IT 服务 (IT Services)：与系统和网络日常维护人员主要关心诸如磁盘空间使用情况、系统负载情况、网络可用性等具体的监控细节不同，一个组织中某些高层领导，例如一个公司的总经理、老板或者运维部门的负责人等，它们可能更关心的是 IT 基础设施“高层”的监控信息。例如，IT 部门或服务商所提供的某个服务在某段时间的可靠性、IT 基础设施中的薄弱环节等其他“高层”IT 信息。以上这些信息的监控，在 Zabbix 系统中都是通过 IT 服务功能来完成的。

Zabbix 系统中的 IT 服务是由层次结构的监控数据表示的，如图 1-10 所示。

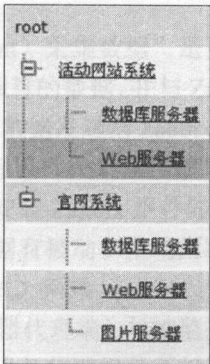


图 1-10 IT 服务层次结构

从图 1-10 可以看出，整个 IT 服务是一个由不同服务组成的层次结构，IT 服务的根是 root，其他服务都是它的子节点。该结构中的每个节点都有状态属性。状态是通过一定的算法计算出来的，并且按照一定的算法传导到它的上级节点。最低级的服务是触发器，它的状态是由该触

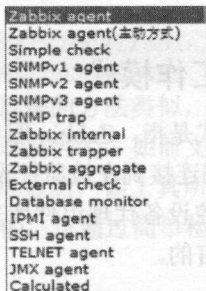
发器的状态所决定的。需要注意的是，“未定义”级别或“信息”级别触发器的状态不会影响 SLA 的计算结果。

1.8 本章小结

本章介绍了什么是 Zabbix 系统以及 Zabbix 系统所具有的特点，并将 Zabbix 与常见的几种开源监控软件做了比较。本章还详细介绍了 Zabbix 独立服务器安装和部署的步骤，并简单介绍了 Zabbix 系统中的基本概念，以使读者对 Zabbix 系统有一个初步的了解。

第2章 数据采集方法介绍

Zabbix 系统支持非常多的监控数据采集方法,如图 2-1 所示的列表就是从 Zabbix 系统的 Web 前端页面中截取出来的,这是一个 Zabbix 系统可以支持的监控数据采集方法列表。当然,列表中所列的监控数据采集方法,也并不是在每一台监控服务器或监控系统中都需要使用。在实际工作环境中,可以根据我们自己的实际工作环境,以及所需要监控的项目特点,选择几种适合于我们自己的数据采集方法。即使在分布式的监控环境中,很多时候,也可能只选用几种适用于我们所需要的数据采集方法,而并不是所有的方法都会用到。接下来,在介绍如何使用 Zabbix 系统监控一台具体主机之前,让我们花一点时间来认识 Zabbix 系统中的这些监控数据采集方法。

A screenshot of a web interface showing a list of Zabbix agent methods. The list is enclosed in a black border and has a dark background with light-colored text. The methods listed are: Zabbix agent, Zabbix agent(主动方式), Simple check, SNMPv1 agent, SNMPv2 agent, SNMPv3 agent, SNMP trap, Zabbix internal, Zabbix trapper, Zabbix aggregate, External check, Database monitor, IPMI agent, SSH agent, TELNET agent, JMX agent, and Calculated.

Zabbix agent
Zabbix agent(主动方式)
Simple check
SNMPv1 agent
SNMPv2 agent
SNMPv3 agent
SNMP trap
Zabbix internal
Zabbix trapper
Zabbix aggregate
External check
Database monitor
IPMI agent
SSH agent
TELNET agent
JMX agent
Calculated

图 2-1 Zabbix 系统中数据采集方法列表

2.1 通过被监控设备代理采集数据

所谓“通过被监控设备代理(agent)采集数据”是指,在被监控设备上运行被监控设备代理(agent)组件,通过该组件收集监控项目的数据,并与 Zabbix 服务器或监控服务器代理(Proxy)通信,主动或被动地将所采集到的监控数据发送给 Zabbix 服务器或其代理的监控数据采集方式。所以,要通过 Zabbix 被监控设备代理组件采集监控数据,则需要在被监控设备上安装并运行 Zabbix 被监控设备代理组件(在编译安装 Zabbix 组件时,使用--enable-agent 编译配置选项)。

通过被监控设备代理采集监控数据有被动和主动两种工作模式,分别对应于图 2-1 所示的 Zabbix agent 和 Zabbix agent(主动方式)两项。

Zabbix 服务器端(或 Zabbix 服务器代理端)与被监控设备代理端之间的数据通信格式采用的是 JSON 格式。实际上,当被监控设备代理组件所封装的监控项目种类(被监控设备代理组件可以采集哪些种类监控项目的数据,是在该组件的源程序中写死的)。不能满足需要的时候,完全可以使用我们所擅长的任何一种高级语言,来开发小的工具或脚本模拟被监控设备代理组件,从而实现我们对于监控的个性化需求。而当通过小工具或脚本模拟被监控设备代理组件,从而实现个性化的监控数据采集需求时,一般来说,使用主动模式要比使用被动模式实现起来

简单和方便。其原因其实也很简单，使用主动模式时，这类小工具不需要以守护进程的方式运行，更不需要在被监控设备上开启额外的服务端□，只需这类小工具或脚本能定时将所采集的数据发送到 Zabbix 服务器或其代理的指定端口上即可。这样，实现起来就非常简单了，甚至可以使用脚本语言（例如 Python）编写脚本，然后通过 crontab 定时任务定时执行该脚本并发送相应数据即可。关于如何编写和使用这样的小工具或脚本，在后续章节中有详细的介绍和说明，并有相应的实例。

从上面的介绍中可以看出，通过 Zabbix 系统所提供的被监控设备代理组件采集监控数据，其所支持的监控项目的种类是相对固定的。这是因为，其收集监控数据的方法是在被监控设备代理组件的源代码里实现的，所以，除非你修改被监控设备代理组件的源代码，并重新编译和安装，否则其所支持的监控项目的种类是相对固定的。当然，不同版本的被监控设备代理组件，其所支持的监控项目的种类是有一些差异的。本书的附录 A，详细列出了 Zabbix 2.0.0 版本下的被监控设备代理组件所支持的监控项目的种类。

为了避免过度消耗 Zabbix 服务器上的内存，Zabbix 系统限定被监控设备代理组件每次发给服务器端（或 Zabbix 服务器代理端）的数据不得大于 128MB。实际上，128MB 大小的数据限制是足够满足我们监控几乎所有监控项目需要的。所以，基本上不需要考虑 Zabbix 系统的这条限制。

2.1.1 被监控设备代理被动工作模式

所谓被监控设备代理被动工作模式是指，Zabbix 服务器端（也可能是服务器端代理）按照监控项目配置所指定的时间间隔，定期连接到在被监控设备上运行的代理（agent）守护进程（默认侦听端口是 10050），并获取被监控设备代理组件所采集的监控数据的工作模式，它是相对于被监控设备代理的主动工作模式而言的。

2.1.2 被监控设备代理主动工作模式

相对于被监控设备代理被动工作模式而言，被监控设备代理的主动工作模式要复杂一些。运行在被监控设备上的被监控设备代理组件（agent）需要首先从 Zabbix 服务器端（或 Zabbix 服务器代理端）获取需要采集数据的监控项目列表及它们的配置信息。而被监控设备代理组件确定它所需要连接的 Zabbix 服务器或 Zabbix 服务器代理的方法是，从其自身的配置文件（zabbix_agentd.conf）中读取 ServerActive 配置项的内容。该配置项的值指定了被监控设备代理组件所需要连接的 Zabbix 服务器（或 Zabbix 服务器代理）的 IP 地址或主机名。而被监控项目列表及其配置信息的读取频率则是由上述配置文件中的 RefreshActiveChecks 配置项来指定的。但是，如果被监控设备代理组件某次从服务器端读取被监控项目列表出错，那么它将在 60s 后重试。

完成监控项目列表及其配置信息的读取后，被监控设备代理组件会根据每个监控项目的配置情况定期采集监控数据，并且在采集到所需要的数据后，立即主动将所采集到的监控数据发送给 Zabbix 服务器或其代理。

由此可见，所谓被监控设备代理主动工作模式是指被监控设备代理会根据监控项目的配置信息，主动采集监控数据，并主动将所采集到的监控数据发送给 Zabbix 服务器或其代理。而被监控设备代理被动工作模式则不然，在被动工作模式下，被监控设备代理并不会主动采集监控数据，且不会主动地将所采集的监控数据发送到 Zabbix 服务器或其代理，而只有当其接收到来

自 Zabbix 服务器或其代理的针对某个监控项目采集数据的请求后，被监控设备代理才会采集相应监控项目的数据，并将所采集到的数据发送给 Zabbix 服务器或其代理。

2.2 简单检查

正如简单检查（Simple Check）这种监控数据采集方法的名字所揭示的那样，这种监控数据采集方法是指 Zabbix 服务器（也包括 Zabbix 服务器代理）自身通过一定的方式或方法检查被监控设备上的网络端口状态或 ICMP 的信息，从而获取监控数据的一种监控数据采集方法。

在 Zabbix 系统的简单检查中，ping 检查需要安装第三方工具包——fping。如果没有安装这个软件包，或者 fping 命令的权限设置不正确，或者 Zabbix 服务器端（或 Zabbix 服务器代理）的配置文件中所指定的 fping 命令路径（通过配置文件中的 FpingLocation 配置项来指定）与 fping 命令的实际安装路径不一致，那么简单检查方法中的 ping 功能（包括 icmping、icmpingloss 和 icmpingsec）都将无法使用。

运行 Zabbix 服务器端和其代理进程的系统用户，通常用户名为 zabbix，其需要对 fping 命令具有可执行权限，而且 fping 命令需要针对 root 用户设置 setuid 权限位。对 fping 命令设置权限的命令如下：

```
shell> chown root:zabbix /usr/local/sbin/fping
shell> chmod 4710 /usr/local/sbin/fping
```

与通过被监控设备代理采集监控数据的方法类似，通过简单检查的方法所能采集监控数据的监控项目的种类也是相对固定的。这是因为，如前文所述，通过简单检查的方法采集监控数据需要使用第三方工具——fping，而很显然，该工具所支持的功能和可传递的参数是相对固定的。表 2-1 列出了 Zabbix 系统所支持的、可通过简单检查的方法采集监控数据的监控项目种类信息。

表 2-1 简单检查类监控项目列表

描述	通过ICMP协议的ping方法检查主机是否存活
关键字语法	icmping[<target>,<packets>,<interval>,<size>,<timeout>]
返回值含义	返回值为0时表示ping失败。一般这也就意味着对应主机不在线或不可用，防火墙或主机禁ping的情况例外。返回值为1时，则表示ping成功，也就意味着对应主机存活
参数说明	target: 被监控主机的IP地址或合法的DNS主机名；packets: 每次连续ping数据包的个数；interval: 相邻两次发送ping数据包的时间间隔，单位为μs；size: 每次发送ping数据包的大小，单位为字节；timeout: 超时时间，单位是μs
备注	例子，如果某个监控项目的关键字为icmping[4]，则表示连续发送4个ping数据包，只要有一个数据包得到正确回应，则该监控项目在此刻所采集到的监控数据就为1
描述	返回丢失包的百分比
关键字语法	icmpingloss[<target>,<packets>,<interval>,<size>,<timeout>]
返回值含义	返回丢失的数据包占已发送的ping数据包的百分数
参数说明	target: 被监控设备的IP地址或合法的DNS主机名；packets: 每次连续发送的ping数据包的个数；interval: 相邻两次发送ping数据包的时间间隔，单位为μs；size: 每次发送ping数据包的大小，单位是字节；timeout: 超时时间，单位是μs
备注	

续表

描述	返回ICMP ping的响应时间
关键字语法	icmppingsec[<target>,<packets>,<interval>,<size>,<timeout>,<mode>]
返回值含义	返回值为被检查主机到Zabbix服务器（或其代理）之间的响应时间
参数说明	target: 被监控设备的IP地址或合法的DNS主机名; packets: 每次连续发送的ping数据包的个数; interval: 相邻两次发送ping数据包的时间间隔, 单位为μs; size: 每次发送ping数据包的大小, 单位是字节; timeout: 超时时间, 单位是μs; mode: 模式, 可选择值有min（最小值）、max（最大值）和avg（平均值）
备注	如果被监控设备的网络不可到达（或者超时），则返回值为0
描述	检查某个服务的TCP端口是否可接受连接
关键字语法	net.tcp.service[service,<ip>,<port>]
返回值含义	若指定的TCP端口可以正常连接, 则返回值为1; 反之则返回值为0
参数说明	service: 可选的值有ssh、ntp、ldap、smtp、ftp、http、pop、nntp、imap、tcp、https和telnet等; IP: 被监控设备所对应的IP地址或合法的DNS主机名; port: 所要检查的TCP端口号, 如果省略, 则使用指定服务或协议的标准端口号
备注	例子: 关键字为net.tcp.service[ftp,,45]的监控项目, 用于检查指定主机上的ftp服务, 但是端口号是45。如果所检查的服务名为tcp, 则端口号必须指定。使用这种方法来检查服务的可用性, 可能会引起被监控设备上对应服务产生额外的日志信息。某些使用加密协议的服务, 如IMAP等, 目前暂不支持使用该方法来检查。当被检查的服务是telnet时, 则Zabbix系统是通过检查主机的响应中是否包括“:”号提示符来判断telnet服务是否可用的
描述	检查服务的性能
关键字语法	net.tcp.service.perf[service,<ip>,<port>]
返回值含义	当返回值为0时, 则表示对应的服务不可用; 否则, 返回Zabbix服务器或其代理连接到对应服务所花的时间, 单位是秒
参数说明	service: 可选的值有ssh、ntp、ldap、smtp、ftp、http、pop、nntp、imap、tcp、https和telnet等; IP: 被监控设备所对应的IP地址或合法的DNS主机名; port: 所要检查服务的端口号, 如果省略, 则使用指定服务或协议的标准端口号
备注	例子: 关键字为net.tcp.service.perf[ssh]的监控项目, 用于检测Zabbix服务器或其代理连接到指定主机上SSH服务所花的时间。如果被检测的服务名是tcp, 则端口号必须指定。某些使用加密协议的服务, 如IMAP等, 目前暂不支持使用该方法来检测。当被检查的服务是telnet时, 则Zabbix系统通过检查主机的响应中是否包括“:”号提示符来判断telnet服务是否可用或是否已经连接上了

- 提示: 1. 何为监控项目的关键字以及它们的作用, 我们在后续章节中有详细介绍。
2. 监控项目关键字中的参数可以省略。但是, 在第一个未省略的参数之前, 如有参数被省略了, 那么分隔该被省略参数的“,”号必须要有。但是, 最右边连续被省略的参数, 可以不需要填写相应的“,”号。
3. 你或许觉得 service 参数的可选值中所列的这些应用层服务, 其在网络层协议上均是基于 TCP 协议的。那么, 既然 Zabbix 系统可以直接检测 tcp 服务的状态和性能, 那为什么还需要单独对 SSH、FTP 等服务进行检测呢? 实际上, 对于像 SSH、FTP、SMTP 等应用层服务, 直接通过 TCP 服务并根据相应端口号也是可以检测的。但是, 某些应用层服务, 对客

户端连接是否成功的响应是有固定特征码的。例如，POP3 服务，对客户端成功连接的响应特征码是“OK”；而 SMTP 服务，对客户端成功连接的响应特征码是“220”。这样，在 service 参数中使用更确切的服务名称，将会使检测的结果更加准确。

4. Zabbix 系统每次执行单项简单检查的最长时间，不会超过 Zabbix 服务器（或 Zabbix 服务器代理）端组件所对应的配置文件中相关配置项所指定的超时时间，因此，表 2-1 中 timeout 参数所指定的超时时间不应超过 Zabbix 服务器（或 Zabbix 服务器代理）端配置文件中所指定的超时时间。

我们知道，在 Zabbix 系统的简单检查中，ping 检查需要使用到第三方命令工具——fping。不难理解，在 Zabbix 系统调用 fping 命令工具时，一般不会也没有必要将 fping 命令工具所支持的所有参数都显式地传递给它，对于一些非关键的、可以使用默认值的参数，Zabbix 系统在调用 fping 命令工具时就不会显式地传递给它。同时，对于某些用户在监控项目的配置中没有显式指定的参数，Zabbix 系统在调用 fping 命令工具时，也可能会使用自身设定默认值作为调用 fping 命令工具时的参数值传递给 fping 命令工具。需要提醒注意的是，fping 命令工具参数的默认值和 Zabbix 系统调用 fping 命令工具时所使用的参数默认值不是一码事。前者是指，Zabbix 系统在调用 fping 命令工具时没有显式地指定某个参数的值，此时 fping 命令工具会使用相应参数的默认值来执行；而后者则是指，当用户在监控项目的配置中没有显式地指定某个参数时，Zabbix 系统会使用自身设定的该参数的默认值来调用 fping 命令工具。表 2-2 列出了这两类默认值的信息。需要说明的是，某些参数的默认值可能会随着安装 fping 工具的平台及版本的不同而不同。

表 2-2 fping命令参数默认值列表

参数	单位	描述	fping 选项	默认值		取值范围	
				fping	Zabbix服务器	最小值	最大值
packets	个	每次检查时发送到主机的数据包个数	-C		3	1	1000
interval	μs	两次成功检查之间的时间间隔	-p	1000		20	没有限制
size	字节	发送的数据包的字节数，x86平台下是56个字节，x86_64平台是68字节	-b	56或68		24	65507
timeout	μs	超时时间	-t	500		50	没有限制

2.3 通过 SNMP 协议采集数据

在 Zabbix 系统可监控的对象中，除了像服务器或各类工作站等，这类用户可以自己自由安装软件的设备之外，还有大量的类似网络交换机、防火墙、路由器、网络存储设备甚至网络打印机等设备，这类设备用户基本上无法在其上安装第三方软件。所以，如果要监控这些类型设备的性能数据，如端口的网络流量、CPU 负载、磁盘的状态等，很显然无法通过被监控设备代理组件来采集数据，而通过简单检查的方法所获得的数据又往往无法满足我们的要求。这个时

候, 通过 SNMP 协议来采集我们感兴趣的数据, 是一种最简便而又最有效的数据采集方法。

从图 2-1 所示的 Zabbix 系统中数据采集方法列表中可以看出, 在 Zabbix 系统中, 可以使用的与 SNMP 协议有关的数据采集方法有 SNMP V1 Agent、SNMP V2 Agent、SNMP V3 Agent 以及 SNMP trap 等四种方法。在具体介绍这些数据采集方法之前, 让我们花一点时间来认识一下什么是 SNMP 协议以及它的相关知识。

2.3.1 SNMP 协议介绍

简单网络管理协议 (Simple Network Management Protocol, SNMP), 是互联网工程工作小组 (IETF, Internet Engineering Task Force) 定义的 Internet 协议簇的一部分。它现在广泛应用于各种网络设备和服务器上, 以使用户能够通过它获取网络设备或服务器上的各类信息。特别是某些网络设备, 例如防火墙、路由器、交换机等, 监控系统要想获取其性能数据, 如 CPU 负载、端口流量等, 通过 SNMP 协议获取是最简便和最有效的方法。

如果打算通过 SNMP 协议来采集监控数据, 那么就要在编译安装 Zabbix 服务器和 Zabbix 服务器代理组件时增加上 SNMP 支持 (即在编译配置时使用 `--with-net-snmp` 配置选项)。另外, 因为 SNMP 协议是通过 UDP 协议而不是 TCP 协议来传输数据的, 而 UDP 是一种非可靠的网络传输协议, 所以通过 SNMP 协议采集监控数据时, 可能存在数据丢失的现象。笔者在网上经常看到有网友抱怨 Zabbix 系统 (其实不仅仅是 Zabbix 系统, 其他任何一款通过 SNMP 协议采集监控数据的监控软件可能都存在这种情况) 在发送数据图时有断图的情况发生, 使用 SNMP 协议采集监控数据可能就是其中原因之一。特别是当网络流量大, 网络非常拥堵的时候, 更容易发生监控数据丢失的现象。

有一系列的规范文件定义了 SNMP 协议。本书不是专门介绍网络管理方面知识的专业书籍, 故以下我们对 SNMP 协议的相关内容不作过深的探讨, 而仅介绍在使用 SNMP 协议采集监控数据时需要熟悉的几个 SNMP 协议方面的概念。如果读者需要更深入地了解 SNMP 协议方面的知识, 请参阅相关的网络管理方面的书籍。

1. 管理信息库 (Management Information Base, MIB)

管理信息库应该不难理解, 像平时我们可能比较熟悉的系统 CPU 负载、系统 CPU 型号、CPU 个数、系统已使用的内存空间大小、系统磁盘状态是否良好、CPU 温度甚至系统风扇当前转速等对象信息, 需要按照一定的规范组织起来并存放, 以供用户或其他服务在需要的时候进行查询或修改。而这些可以通过 SNMP 协议进行管理的对象的集合就构成了管理信息库。

管理信息库中的所有对象信息, 是分层次按树状结构组织起来的。虽然设备生产商可以根据需要, 扩充自己所生产设备的管理信息库中对象的内容, 但是 SNMP 协议标准所要求的标准对象必须要在任何一个管理信息库中实现, 且这些标准对象在管理信息库的对象树中的位置是相对固定的。例如, 表示网络设备拥有网络接口 (包括交换机的端口数或服务器的网卡等) 数量的对象 `ifNumber` 在对象树中的位置如图 2-2 所示。

当管理信息库中的所有对象信息按照图 2-2 所示分层次、按树状结构组织起来后, 如果需要查询某台开启了 SNMP 服务的网络设备所拥有的网络接口数量, 则只需查询管理信息库中的 `ifnumber` 叶子对象的值即可。

2. 对象标识符 (Object Identifier, OID)

如果需要在图 2-2 所示的树形 MIB 中查找某个对象并取得该对象的值, 就需要使用某种方

法来定位和标识 MIB 库中的对象。在 SNMP 协议中，使用对象标识符——OID 来标识管理信息库中的对象。OID 表示的方法为：将从管理信息库的根开始到指定对象所经过的各层次对象的位置符用点号（“.”）从左到右连接起来，所形成的字符串就是对应对象的标识符。例如，如果需要标识图 2-2 中的 ifnumber 对象，则只需将管理信息库中，从根开始到 ifnumber 节点所经过的各层次对象的位置标识数字从左到右连接起来，形成一串用点号分隔的数字：1.3.6.1.2.1.2.1，这就是对象 ifnumber 的 OID 标识符。上述的这种对象标识方法是对象标识符的数字表示方法，当将各层次对象的名称用点号从左到右连接起来，形成如 .iso.org.dod.internet.mgmt.mib-2.interface.ifnumber 一串用字符所表示的路径，这便是 OID 的名称标识方法。

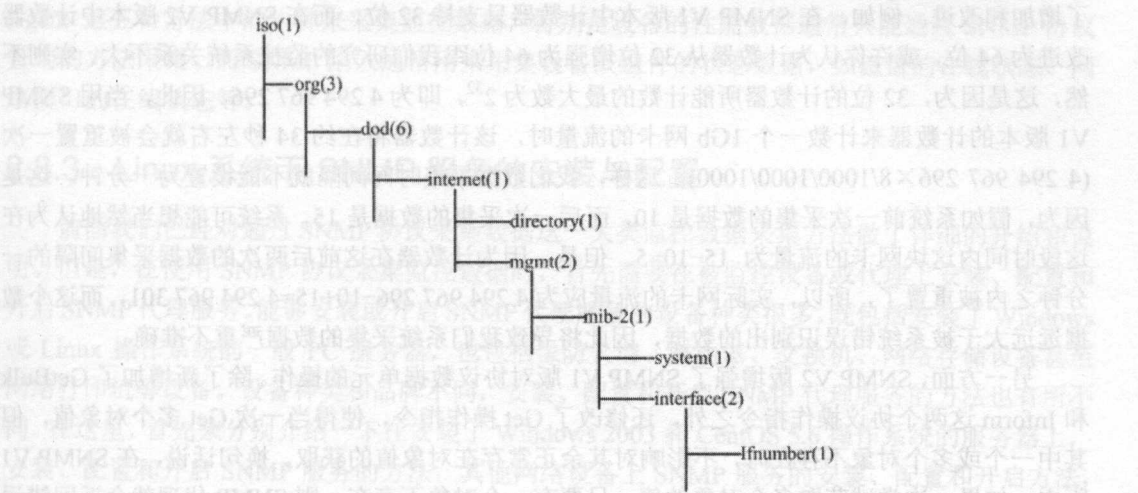


图 2-2 MIB 对象结构图

上述这两种对象的标识方法都是从管理信息库的根开始记录和标识的，由这两种标识方法所得出的对象标识符称之为对象的绝对标识符，即对象的绝对 OID。当对象的标识符未从管理信息库的根开始，而是以其某个父节点为基准，将从其父节点开始到该节点所经过的各层次对象的位置标识数字（或名称）从左到右用点号连接起来所形成的对象标识符，就称之为对象的相对标识符。例如，当以 internet 节点为基准来标识 ifnumber 对象时，其相对标识符就应为：2.1.2.1。

不管是对象的绝对 OID 还是相对 OID，我们在实际监控中使用的都不是很多。实际上，我们在 Zabbix 系统中看到的或者说使用的最多的还是类似于 SNMPv2-MIB::sysName.0 形式的 OID。这是 OID 的另一种标识方法，其所表示的对象是 SNMPv2 信息库中 sysName 对象下第一节点的对象，即设备的系统全称。

2.3.2 SNMP 协议版本

SNMP 协议常用的版本有三个，分别对应于 SNMP V1、SNMP V2 和 SNMP V3。相对应于这三个 SNMP 协议版本，在 Zabbix 系统中对应地使用 SNMP 协议采集监控数据的方法分别是：SNMP V1 Agent、SNMP V2 Agent 和 SNMP V3 Agent。

SNMP V1 是 SNMP 协议的第一个版本，它指定了四种核心协议数据单元(Protocol Data Unit, PDU)。分别是 GET，用来获取指定对象的信息；GETNEXT，用来连续获取指定的多个对象的信息；SET，用来设置一个管理对象的信息；TRAP，当管理对象的信息发生变化时，主动向管

理站（在我们的监控体系中就是指 Zabbix 服务器或 Zabbix 服务器代理）报告这种变化。SNMP V1 只使用了一种安全策略，即团体名（community）。团体名和密码类似，即只有能够提供出正确团体名的管理站（例如我们的 Zabbix 服务器）才能被接受查询或设置。而在 SNMP V1 版本中，管理站和 SNMP 代理之间相互发送的 PDU 都是明文的，这就很容易造成 SNMP 管理站与代理之间的通信被截取。也正是因为这个原因，在实际应用中，现在一般很少使用 SNMP V1 进行监控数据的采集，除非我们所监控的主机是比较老的设备，其不支持其他版本的 SNMP 协议，否则至少应该使用 SNMP V2 进行监控数据的采集。

SNMP V2 是 SNMP V1 的演进版。它在性能、安全和协议操作等方面对 SNMP V1 版本做了增加和改进。例如，在 SNMP V1 版本中计数器只支持 32 位，而在 SNMP V2 版本中计数器改进为 64 位。或许你认为计数器从 32 位增强为 64 位跟我们研究的监控系统关系不大。实则不然，这是因为，32 位的计数器所能计数的最大数为 2^{32} ，即为 4 294 967 296。因此，当用 SNMP V1 版本的计数器来计数一个 1Gb 网卡的流量时，该计数器将在约 34 秒左右就会被重置一次（ $4\,294\,967\,296 \times 8 / 1000 / 1000 / 1000$ ）。这样，我们数据采集时间间隔就不能设置为一分钟。这是因为，假如系统前一次采集的数据是 10，而后一次采集的数据是 15，系统可能想当然地认为在这段时间内这块网卡的流量为 $15 - 10 = 5$ 。但是，因为计数器在这前后两次的数据采集间隔的一分钟之内被重置了，所以，实际网卡的流量应为 $4\,294\,967\,296 - 10 + 15 = 4\,294\,967\,301$ ，而这个数据远远大于被系统错误识别出的数据，因此将导致我们系统采集的数据严重不准确。

另一方面，SNMP V2 版增强了 SNMP V1 版对协议数据单元的操作。除了新增加了 GetBulk 和 Inform 这两个协议操作指令之外，还修改了 Get 操作指令，使得当一次 Get 多个对象值，但其中一个或多个对象不存在时，不影响对其余正常存在对象值的获取。换句话说，在 SNMP V1 版中，如果一次尝试获取多个对象的值，只要有一个对象不存在，则 SNMP 代理就会返回错误的提示，并且余下对象的查询都将不会继续进行，即使余下的对象在管理信息库中是真实存在的。

SNMP V3 版本在 SNMP V2 版本的基础上增强了安全性和远程配置。主要包括：信息完整性校验，以保证数据包在传送的过程中没有被篡改；放弃了前两个版本中仅依赖团体名认证的方式，而提供了用户和密码认证方式；增加了数据包加密，在 SNMP V3 版本中，协议数据单元中除了目的地址之外，其余数据在传输过程中都是被加密的，这样避免了被未经授权的用户进行窥探和截取。

不管是使用 SNMP V1、SNMP V2 还是使用 SNMP V3 版本的 SNMP 协议采集监控数据，其本质上都是利用 SNMP 协议来采集监控数据。所以，以下我们将 SNMP V1 Agent、SNMP V2 Agent 和 SNMP V3 Agent 这三种 Zabbix 系统里的监控数据采集方法放在一起讨论，只要明白了上述三种利用 SNMP 协议采集监控数据的方法中的一种，其他两种利用 SNMP 协议采集监控数据的方法在原理和方法上都是一样的，没有本质上的区别。

但是，利用 SNMP 协议进行监控数据采集的第四种方法——SNMP 陷入（即 SNMP trap）可能与前面所述的三种利用 SNMP 协议采集监控数据的方法略有区别。前面讨论的 SNMP V1 Agent、SNMP V2 Agent 和 SNMP V3 Agent 这三种利用 SNMP 协议采集监控数据的方法，都是 Zabbix 服务器或者 Zabbix 服务器代理根据监控项目的配置，定期通过 SNMP 协议查询被监控设备上的对象信息，它是一种被动的数据采集方式。而 SNMP 陷入则是指当 SNMP 代理在被监控设备上的某个条件发生时，根据其自身配置向指定的地址和端口发送相应的 SNMP 信息。与 SNMP 协议的另外三种监控数据采集方法不同的是，SNMP 陷入是一种主动的监控数据采集方

式,即由被监控设备主动向 Zabbix 服务器或 Zabbix 服务器代理发送要采集的数据。需要注意的是,并不是所有支持 SNMP 协议采集监控数据的监控项目都可以使用 SNMP 陷入来采集监控数据。而具体哪些监控项目可以使用 SNMP 陷入的方法来采集监控数据,则是由被监控设备上的 MIB 库决定的。一般来说,像硬件状态数据,比如磁盘状态数据、网卡状态数据等,都是可以通过 SNMP 陷入来采集的。通常,支持 SNMP 陷入的对象种类并不是很多,因而可采用 SNMP 陷入的方法采集监控数据的监控项目的种类通常也不是很多,而通过 SNMP 协议查询的方法则可以查询到设备中 MIB 库里所定义的所有对象的信息。因而,在绝大多数的情况下,我们都是使用 SNMP 协议查询的方法,即使用 SNMP V1 Agent、SNMP V2 Agent 和 SNMP V3 Agent 这三种方法中的一种来采集监控数据,特别是设备的性能数据通常只能通过 SNMP 协议查询的方法采集。而 SNMP 陷入通常用来采集设备或组件的状态数据,如磁盘的在线状态、网络接口的在线状态等。

2.3.3 Linux 系统下 SNMP 服务的安装与配置

前面我们已经对通过 SNMP 协议采集数据这一大类监控数据采集方法做了详细的介绍和说明。但是,在使用 SNMP 协议采集监控数据时,首先需要在被监控设备或代理上安装、配置和开启 SNMP 代理服务。能够安装或开启 SNMP 代理服务的设备种类很多,既包括安装了 Windows 或 Linux 操作系统的一般 PC 服务器,也包括像防火墙、路由器、交换机、网络存储设备甚至网络打印机等设备。设备种类和品牌不同,安装、配置和开启 SNMP 代理服务的方法也有所不同。在这里,首先来分别介绍一下在安装了 Windows 2003 和 CentOS 5.6 操作系统的服务器上,安装、配置和开启 SNMP 服务的方法,其他网络设备上 SNMP 服务的安装、配置和开启方法,可参考相应的产品用户手册。

在 Linux 系统下安装 SNMP 服务还是比较简单的,首先使用 yum 来安装 net-snmp 软件包:

```
shell > yum install net-snmp
```

根据提示输入 y, 下载并安装 net-snmp 软件包及其所依赖的软件包即可。

下面这两个软件包里包含了有关 SNMP 服务的一些命令行工具,不需要在每台被监控设备及其代理上安装。这里选择安装这两个软件包,是因为在下面测试 SNMP 服务是否安装配置正确时需要这两个软件包里的工具。

```
shell> yum install net-snmp-utils
```

```
shell> yum install net-snmp-devel
```

安装好 net-snmp 软件包后,使用下列命令将 SNMP 服务器添加到开机自动启动服务列表中,以便 SNMP 服务能在每次重启操作系统时自动启动:

```
shell> chkconfig snmpd on
```

在完成安装 net-snmp 软件包后,系统会生成一个原始 SNMP 服务配置文件。很显然,按理需要针对这个原始配置文件进行修改,包括修改团体名、接受查询的客户端 IP 或合法的 DNS 主机名、允许客户查询 MIB 库的范围等,并且依照修改后的配置文件开启的 SNMP 服务才能正常使用。然而,系统所提供的原始文件里有很多关于 SNMP 配置的说明,同时该文件中还有很多配置项在实际监控中很少用到。限于篇幅原因,在这里,我们提供一段简化的 SNMP 服务配置文件内容,读者只需将这里所提供的简化的配置信息,根据实际工作环境需要稍加修改并替换掉系统为我们生成的原始配置文件里的内容,即可满足绝大多数情况下监控的需要。

```
1. rouser snmpuser
2. rocommunity snmp@domain.com 192.168.5.0/24
3. rocommunity snmp@domain.com 127.0.0.1
```



```

4. group notConfigGroup v1 notConfigUser
5. group notConfigGroup v2c notConfigUser
6. view systemview included .1.3.6.1.2.1.1
7. view systemview included .1.3.6.1.2.1.25.1.1
8. access notConfigGroup "" any noauth exact all none none
9. view all included .1
10. trapsink 192.168.5.139:162 public
11. trap2sink 192.168.5.140:162 snmp@domain.com
12. authtrapsenable 1

```

下面,我们来逐条介绍一下上面这段配置内容的含义。**rouser snmpuser**: 配置一条 SNMP V3 的用户信息,用户名为 **snmpuser**,可根据需要进行修改。需要注意的是,要添加 SNMP V3 用户,并不仅仅是简单地修改 SNMP 服务的配置文件,手动添加上类似于上面这条记录就可以的,我们还需要在系统中创建相应的用户。在 CentOS 系统中,当安装了 **net-snmp** 软件包后,系统会给我们提供一个名叫 **net-snmp-config** 的命令工具。可以使用这个命令工具来创建 SNMP V3 用户,其命令格式如下:

```
shell> net-snmp-config --create-snmpv3-user -ro -A snmp@domain.com -a MD5/SHA snmpuser
```

上面这条命令中的 **-ro** 选项表示所创建的用户只具有只读权限,如果不带 **-ro** 选项,则所创建的用户具有读写权限; **-A** 选项后面的 **snmp@domain.com** 表示用户密码; **-a** 选项表示加密方法,可以选择 MD5 或 SHA。

第 2 行和第 3 行配置了从 192.168.5.0/24 网段和 127.0.0.1 主机访问该 SNMP 服务需使用的团体名为 **snmp@domain.com**,其适用于通过 SNMP V2 和 SNMP V1 版本的协议进行访问。第 4 行和第 5 行则表示,将所有没有使用团名的,并且使用 SNMP V2 和 SNMP V1 版本协议进行的访问归为 **notConfigGroup** 组。第 6、7 行定义了一个视图 **systemview**。第 8 行的配置表示,所有属于 **notConfigGroup** 组的访问都将被拒绝。第 9 行定义了一个视图 **all**。

第 10 行和第 11 行配置了两条 SNMP 陷入,其中第 10 行适用于 SNMP V1 版本,第 11 行适用于 SNMP V2 版本。这两行配置中所配置的 IP 地址,是 SNMP 陷入接收服务器的 IP 地址,在我们的 Zabbix 系统中,这个 IP 地址也就是 Zabbix 服务器(在网络管理术语中,称为管理站。在下文中,我们有时候也会称其为管理站。读者只需知道,这里所说的管理站与 Zabbix 服务器是同一台服务器即可)的 IP 地址。而 SNMP 陷入服务的默认端口号是 162,也就是 **snmptrapd** 进程所侦听的端口号。这两行中的最后一列, **public** 和 **snmp@domain.com** 是团体名。需要注意的是,这里所配置的团体名要与管理站上 **snmptrapd.conf** 文件里所配置的团体名保持一致,否则陷入时认证就会失败。

现在,SNMP 配置文件已经修改好了,接下来保存并启动 SNMP 服务,并测试配置是否正确。

```

shell> service snmpd start          #启动 SNMP 服务
shell> service snmpd stop          #关闭 SNMP 服务

```

在 Zabbix 服务器上,执行下列命令,以检查 SNMP 服务运行是否正常:

```

# 使用 SNMP v1 版本协议验证 SNMP 服务运行是否正常。
shell> snmpstatus -v 1 -c snmp@domain.com 192.168.5.139

# 使用 SNMP v2 版本协议验证 SNMP 服务运行是否正常。
shell> snmpstatus -v 2c -c snmp@domain.com 192.168.5.139

# 使用 SNMP v3 版本协议验证 SNMP 服务运行是否正常。
shell> snmpstatus -v 3 -u snmpuser -a MD5 -A "snmp@domain.com" -l authNoPriv 192.168.5.139

```

以上三条命令如果执行成功，均返回类似如下信息：

```
[UDP: [192.168.5.139]:161]=>[Linux vm2 2.6.18-194.el5 #1 SMP Fri Apr 2 14:58:35 EDT 2010 i686] Up: 0:00:15.19 Interfaces: 3, Recv/Trans packets: 0/26710 | IP: 29422/264531 interface is down!
```

当执行上述命令时，如果服务器返回如“Timeout: No Response from 192.168.5.139”的错误信息，则说明可能对端的 SNMP 服务没有正常启动，或者是对端 SNMP 服务所配置的团体名与我们查询时使用的团体名不一致，再不然就是管理站与对端之间的网络通信被防火墙、路由器等网络设备或服务的策略给阻断了。

当有意将 SNMP 服务停掉时，从上面配置文件 trapsink 或 trap2sink 配置项所指定的服务器系统日志里会看到如下的 SNMP 陷入日志信息：

```
Jan 16 10:05:39 vm2 snmptrapd[7183]: 2014-01-16 10:05:39 sl.test138.com [UDP: [192.168.5.139]:33069]: DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (1128) 0:00:11.28 SNMPv2-MIB::snmpTrapOID.0 = OID: NET-SNMP-AGENT-MIB::nsNotifyShutdown SNMPv2-MIB::snmpTrapEnterprise.0 = OID: NET-SNMP-MIB::netSnmpNotificationPrefix.
```

可以使用下列命令，查询被管理设备上 MIB 库中指定对象的当前信息：

```
# SNMP V1 版本协议查询方法
shell> snmpwalk -v 1 -c snmp@iwgame.com 192.168.5.139 .1 |more

# SNMP V2 版本协议查询方法
snmpwalk -v 2c -c snmp@iwgame.com 192.168.5.139 .1 |more

# SNMP V2 版本协议查询方法
shell> snmpwalk -v 3 -u snmpuser -a MD5 -A "net-snmp@domain.com"
-l authNoPriv 192.168.5.139
```

以上三条命令如果执行成功，均会返回如下信息：

```
SNMPv2-MIB::sysDescr.0 = STRING: Linux vm2 2.6.18-194.el5 #1 SMP Fri Apr 2 14:58:35 EDT 2010 i686
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (256059) 0:42:40.59
SNMPv2-MIB::sysContact.0 = STRING: root@localhost
SNMPv2-MIB::sysName.0 = STRING: vm2
SNMPv2-MIB::sysLocation.0 = STRING: Unknown
SNMPv2-MIB::sysORLastChange.0 = Timeticks: (2) 0:00:00.02
SNMPv2-MIB::sysORID.1 = OID: SNMPv2-MIB::snmpMIB
SNMPv2-MIB::sysORID.2 = OID: TCP-MIB::tcpMIB
SNMPv2-MIB::sysORID.3 = OID: IP-MIB::ip
SNMPv2-MIB::sysORID.4 = OID: UDP-MIB::udpMIB
SNMPv2-MIB::sysORID.5 = OID: SNMP-VIEW-BASED-ACM-MIB::vacmBasicGroup
.....
.....
```

当执行上述命令时，如果服务器返回如“Timeout: No Response from 192.168.5.139”的错误信息，则说明可能对端的 SNMP 服务没有正常启动，或者是对端 SNMP 服务所配置的团体名与我们查询时使用的团体名不一致，再不然就是管理站与对端之间的网络通信被防火墙、路由器等网络设备或服务的策略给阻断了。

之所以在上面很详细地介绍了 snmpstatus 和 snmpwalk 命令工具的用法，是因为通过 SNMP 协议采集监控数据的方法是监控系统中最常用的监控数据采集方法，而在实际维护监控系统的工作中，通过 SNMP 协议无法正常采集到预期的监控数据又是最常见的故障之一。此时，就有必要借助 snmpstatus 和 snmpwalk 等命令工具来诊断故障原因。而在 Linux 系统下，除了上面介绍的这两个命令工具之外，与 SNMP 服务调试和查询相关的常见命令工具还有：snmpget、

snmpgetnext 和 snmpbulkwalk 等, 限于篇幅的原因, 在此就不一一介绍了, 有兴趣的读者可以查看这些命令的在线手册和相关参考资料。

2.3.4 Zabbix 服务器上的 SNMP 陷入配置

在前面, 我们提到并在被监控设备或代理上配置了 SNMP 陷入。与 Zabbix 服务器可以直接支持 SNMP 查询不一样, 对于 SNMP 陷入, 需要在 Zabbix 服务器 (或其代理服务器) 上配置并启动接收 SNMP 陷入事件的服务组件, Zabbix 系统才能使用 SNMP 陷入的方法采集监控数据。

1. Zabbix 系统中 SNMP 陷入接收流程

Zabbix 服务器或其代理无法直接接收被监控设备上的 SNMP 服务代理发送的 SNMP 陷入信息, 它需要借助一款第三方插件才能完成这部分工作。以下是 Zabbix 系统中 SNMP 陷入信息的接收流程:

- ① 运行在 Zabbix 服务器或 Zabbix 代理服务器上的 snmptrapd 进程, 接收到来自被监控设备或其代理发送过来的 SNMP 陷入信息。
- ② snmptrapd 进程将接收到的陷入信息传送给 SNMP 陷入转译器 (SNMPTT、SNMP Trap Translator) 或用 Perl 编写的陷入接收器 (Zabbix 软件包里附带这个接收器程序)。
- ③ SNMPTT 或 Perl 接收器解析接收到的陷入信息, 并按照一定格式进行格式化, 之后将格式化后的陷入信息写入到指定的陷入信息文件中。
- ④ Zabbix 服务器或其代理的相关进程读取并解析这个陷入信息文件里的内容。
- ⑤ Zabbix 服务器或其代理的相关进程, 从每条陷入信息中分离出发送陷入信息的被监控设备的 IP 地址或主机名。
- ⑥ Zabbix 系统从系统所配置的所有主机中, 找出主机名或 IP 地址与从陷入信息中分离出来的 IP 地址或主机名相一致的主机。并将该主机上所配置的监控项目与陷入信息进行正则匹配。如果成功匹配到监控项目, 则将对应的监控项目最新监控数据设置成从陷入信息中分离出来的数据; 如果在对应主机上没有匹配到任何监控项目, 但是在该主机上配置了关键字为 “snmptrap.fallback” 的监控项目, 则系统将从陷入信息中分离出来的数据设置为该监控项目最新监控数据。
- ⑦ 如果 Zabbix 系统以从陷入信息中分离出来的 IP 地址或主机名, 在系统所配置的所有主机中, 没有找到对应的主机, 或者虽然找到了对应主机, 但是从对应主机上未能找到匹配的监控项目, 且也没有关键字为 “snmptrap.fallback” 的监控项目, 则 Zabbix 系统会记录一条没有匹配到陷入信息的日志信息。

2. SNMPTT 安装配置

通过上面的介绍, 我们知道, Zabbix 系统接收 SNMP 陷入信息需要与 snmptrapd 服务 (该服务是 net-snmp 软件包自带的服务) 一起工作。snmptrapd 服务通过其内嵌的机制——Perl 脚本或 SNMPTT 将接收到的 SNMP 陷入信息进行格式化以供 Zabbix 服务器 (或服务器代理) 进程使用。如果使用 Perl 脚本进行 SNMP 陷入信息转译, 则要求在编译 net-snmp 软件时显式地添加内嵌 Perl 支持, 即在编译 net-snmp 软件包时使用 --enable-embedded-perl 编译选项。而我们在介绍安装 net-snmp 包时是使用 yum 方式安装, 默认情况下使用二进制包方式安装的 net-snmp 软件包是不支持内嵌 Perl 语言的 (在 net-snmp 5.4 以后版本的二进制包是支持内嵌 Perl 语言的)。所以, 我们在这里介绍使用 SNMPTT 进行 SNMP 陷入转译的方法。

SNMP 陷入转译器 (SNMP Trap Translator, SNMPTT) 是用 Perl 语言编写的供 net-snmp 或 ucd-snmp 软件包中的 snmptrapd 进程使用的 SNMP 陷入信息处理器。SNMP 陷入转译器的主要作用, 是将 snmptrapd 进程所捕获的陷入信息按照指定的格式进行转换, 并将其保存成文件形式输出到系统日志甚至存储到 SQL 数据库中。下面就来介绍一下 SNMPTT 组件的安装和配置方法。

SNMPTT 组件需要使用 SNMP 及 IniFiles 组件中的 Perl 模块, 所以需要首先安装 net-snmp-perl 以及 perl-Config-IniFiles 软件包。

```
shell> yum install net-snmp-perl
```

perl-Config_IniFiles 软件包在 CentOS 的默认 yum 源里不存在, 而如果使用源码包编译安装, 软件包之间相互依赖关系比较复杂。所以, 在这里使用 epel yum 源来安装这个软件包。首先来配置一下 epel yum 源。创建 /etc/yum.repos.d/epel.repo 文件, 并将下面内容复制进该文件后保存。

```
shell> vi /etc/yum.repos.d/epel.repo
```

```
1. [epel]
2. name=Extra Packages for Enterprise Linux 5 - $basearch
3. mirrorlist=http://mirrors.fedoraproject.org/mirrorlist?repo=epel-5&arch=$basearch
4. failovermethod=priority
5. enabled=1
6. gpgcheck=1
7. gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL
```

配置好 yum 源后, 就可以使用下面命令安装 perl-Config-IniFiles 软件包了:

```
shell> yum install perl-Config-IniFiles
```

接下来, 按照下面的步骤安装配置 SNMPTT 软件包:

下载 snmptt_1.4 源码包

```
shell> wget http://downloads.sourceforge.net/project/snmptt/snmptt/snmptt_1.4/snmptt_1.4.tgz?r=ats=1389880477&use_mirror=superb-dca2
```

```
shell> tar -zxvf snmptt_1.4.tgz # 解压软件包
```

```
shell> cd snmptt_1.4
```

将 snmptt 文件复制到 /usr/sbin 目录下, 并设置成可执行文件

```
shell> cp snmptt /usr/sbin/
```

```
shell> chmod +x /usr/sbin/snmptt
```

将 snmptthandler 文件复制到 /usr/sbin 目录下, 并设置成可执行文件

```
shell> cp snmptthandler /usr/sbin/
```

```
shell> chmod +x /usr/sbin/snmptthandler
```

将 snmptt 配置文件 snmptt.ini 和 snmptt.conf 文件复制到 /etc/snmp 目录下

```
shell> cp snmptt.ini /etc/snmp/
```

```
shell> cp examples/snmptt.conf.generic /etc/snmp/snmptt.conf
```

创建 /var/log/snmptt 和 mkdir /var/spool/snmptt 目录

```
shell> mkdir /var/log/snmptt
```

```
shell> mkdir /var/spool/snmptt
```

创建用户并将上述目录的用户属组设置为新建的用户

```
shell> useradd -s /sbin/nologin snmptt
```

```
shell> chown snmptt:snmptt /var/spool/snmptt
```

修改 SNMPTT 配置文件 snmptt.ini 中下列各项

```
shell> vi /etc/snmp/snmptt.ini
```

修改 mode = standalone 配置项为 mode = daemon;

修改 date_time_format = 配置项为 date_time_format = %H:%M:%S %Y/%m/%d;

修改 log_system_enable = 0 配置项为 log_system_enable = 1;

修改 unknown_trap_log_enable = 0 配置项为 unknown_trap_log_enable = 1, 并确认下面

所列的两项配置是否正确。

```
log_file = /var/log/snmpd/snmpd.log
log_enable = 1
```

3. snmptrapd 服务配置

首先, 需要按照 2.3.3 节所介绍的方法, 在 Zabbix 服务器或其代理服务器上安装 net-snmp 软件包并做相应配置。在安装完 net-snmp 软件包以后, 系统不会自动生成 snmptrapd 进程所需要的配置文件 snmptrapd.conf。所以, 需要使用 snmpconf 命令工具来生成一个 snmptrapd.conf 文件模板。snmpconf 命令工具的使用比较简单, 只要按照提示回答一些提问之后, snmpconf 即可为我们生成 snmptrapd 进程所需要的配置文件模板。在此, 我们提供一个简化的配置文件内容, 读者只需创建 /usr/share/snmp/snmptrapd.conf 文件, 并将下面这段内容复制过去即可:

```
1. donotfork yes
2. pidfile /var/run/snmptrapd.pid
3. printeventnumbers yes
4. traphandle default /usr/sbin/snmpthandler
5. ignoreauthfailure yes
6. authcommunity execute,log,net snmp@domain.com
```

接下来, 因为 SNMPD 是基于数字 OID 来匹配/etc/snmp/snmpd.conf 文件里内容, 以确定接收到了哪种陷入信息, 并将陷入信息转化成相应格式的。但是, 在默认情况下, snmptrapd 进程会自动将所接收到的陷入信息中的 OID 转化成类似于 SNMPv2-MIB::coldStart 这种简化的字符串形式。而这种简化的字符串形式 OID 与 snmpd.conf 文件中所配置的数字形式的 OID 是无法匹配的, 故 SNMPD 服务将无法识别。因此, 需要修改 snmptrapd 进程的启动脚本来解决这一问题。

```
shell> vi /etc/init.d/snmptrapd
```

将 snmptrapd 文件中的 “OPTIONS="-Lsd -p /var/run/snmptrapd.pid"” 行修改成 “OPTIONS="-Lsd -On -p /var/run/snmptrapd.pid"” 并保存。

好了, 经过上面的安装配置和准备, 已经完成了 SNMP 陷入服务的安装和配置。接下来, 需要重启 SNMP 及相关的服务并检查服务工作是否正常。

```
shell> service snmpd start/restart
shell> service snmptrapd start/restart
shell> service snmpd start/restart
```

如果上述三个服务启动都是正常的, 则可以通过下面命令来检查所配置的 SNMP 陷入服务是否工作正常。在本机或其他主机上发送一条陷入信息到 Zabbix 服务器的陷入服务端口上, 并检查 SNMPD 服务是否能在指定的文件里输出我们所预期的陷入信息。

```
shell> snmptrap -v 2c -c snmp@domain.com 192.168.5.139:162 "" .1.3.6.1.6.3.1.1.5.3
```

执行上述陷入命令之后, 如果在 /var/log/snmpd/snmpd.log 文件里有如下的信息输出, 则说明我们所配置的 SNMP 陷入和 SNMPD 服务工作正常:

```
15:05:36 2014/01/17 .1.3.6.1.6.3.1.1.5.3 Normal "Status Events" source.iwgame.
tec - Link down on interface $1. Admin state: $2. Operational state: $3.
```

2.3.5 Windows 2003 下 SNMP 服务的安装与配置

相对于 Linux 系统下 SNMP 服务的安装与配置, Windows 2003 下 SNMP 服务的安装与配置要简单得多。首先, 需要安装 SNMP 组件。依次选择 “开始” → “设置” → “控制面板” 菜单, 并单击控制面板里的 “添加或删除程序” 图标, 然后选择 “添加/删除 Windows 组件(A)” 选项卡, 如图 2-3 所示。

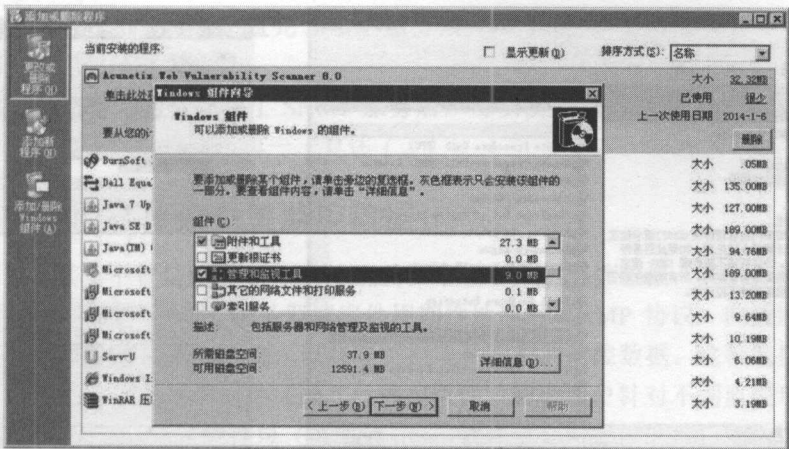


图 2-3 Windows 2003 组件向导

在“Windows 组件向导”界面的“组件”列表框里选中并双击“管理和监视工具”组件，系统会弹出“管理和监视工具”窗口。在“管理和监视工具”窗口中选中“简单网络管理协议(SNMP)”子组件，并单击“确定”按钮，如图 2-4 所示。

在“管理和监视工具”窗口单击完“确定”按钮后，系统将会返回到前一个窗口，即图 2-3 所示的“Windows 组件向导”窗口。在“Windows 组件向导”窗口里单击“下一步”按钮，系统将会安装 SNMP 服务所需要的文件，如图 2-5 所示。

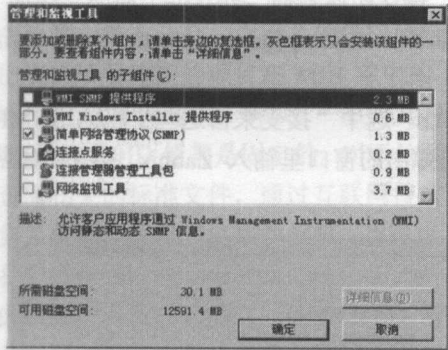


图 2-4 Windows 2003 管理和监视工具窗口

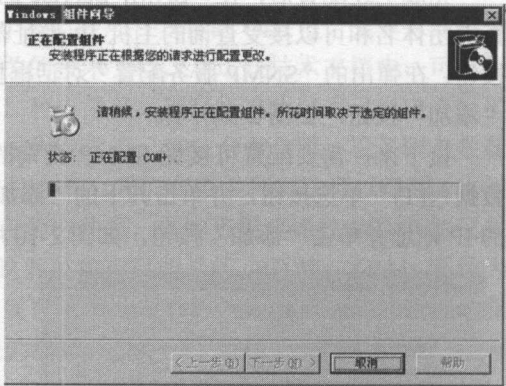


图 2-5 Windows 2003 组件向导

系统安装完所添加组件的文件后，单击“完成”按钮，并退出“添加或删除程序”窗口，完成 SNMP 协议组件的安装。接下来，需要配置 SNMP 服务。依次选择“开始”→“程序”→“管理工具”→“服务”菜单，系统会弹出“服务”窗口，如图 2-6 所示。

在“服务”窗口里找到并选中“SNMP Service”服务，并单击右键，在系统所弹出的右键菜单中单击“属性”菜单项，系统会弹出“SNMP Service 的属性”窗口。在“SNMP Service 的属性”窗口里选择“陷阱”选项卡，如图 2-7 所示。

在“团体名称”栏输入我们之前所配置的团体名，如 snmp@domain.com，然后单击“添加到列表”按钮。单击“陷阱目标”下的“添加”按钮，在弹出的“SNMP 服务配置”对话框里输入 Zabbix 服务器的 IP 地址或合法的 DNS 主机名并单击“添加”按钮，如图 2-8 所示。

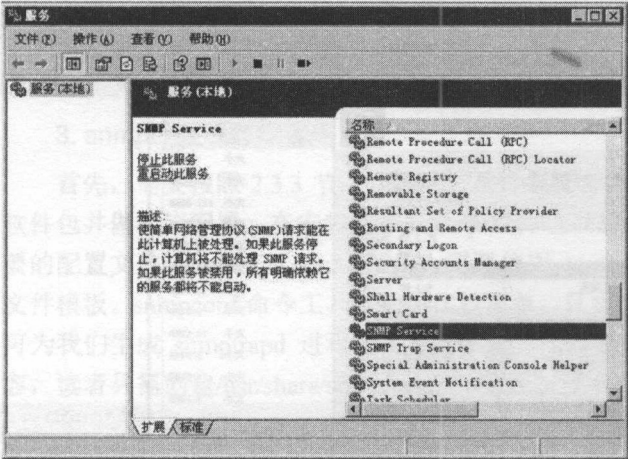


图 2-6 Windows 2003 服务窗口

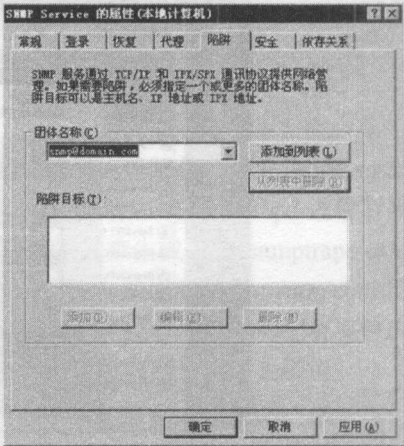


图 2-7 SNMP Service 的属性窗口

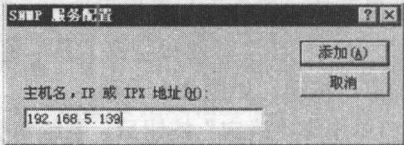


图 2-8 Windows 2003 SNMP 服务配置对话框

完成上述配置后，在“SNMP Service 的属性”窗口中切换到“安全”选项卡，配置 SNMP 查询团体名和可以接受查询的主机 IP 地址范围。单击“接受团体名称”列表框下面的“添加”按钮，在弹出的“SNMP 服务配置”对话框里输入团体名称，如 snmp@domain.com 等，并单击“添加”按钮，如图 2-9 所示。

接下来，需要配置可接受 SNMP 查询的主机 IP 地址。选中“接受来自这些主机的 SNMP 数据包(T)”单选按钮，并单击其下的“添加”按钮，在弹出的窗口里输入 Zabbix 服务器所对应的 IP 地址并单击“添加”按钮，如图 2-10 所示。

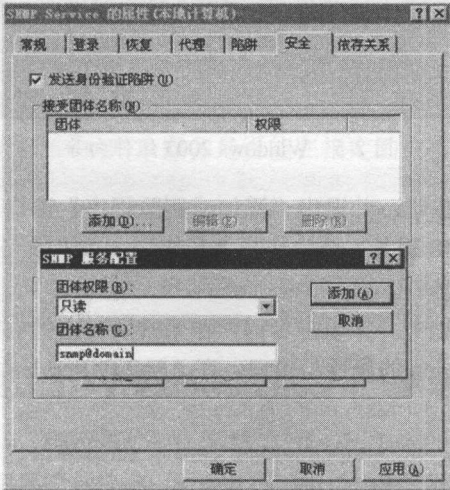


图 2-9 配置 SNMP 团体名

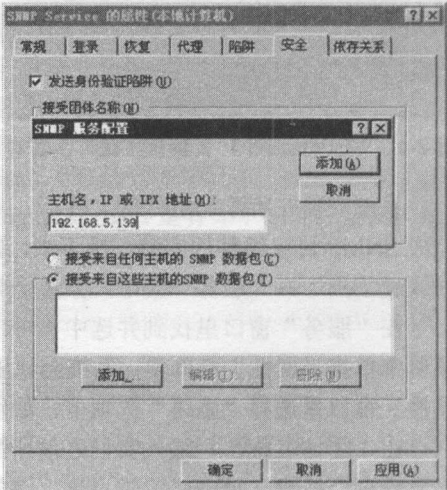


图 2-10 配置 SNMP 服务安全性窗口

完成可接受 IP 地址的配置后，单击“SNMP Service 的属性”窗口上的“应用”和“确定”

按钮以完成 SNMP 服务的配置。配置完 SNMP 服务后，在“服务”窗口里重启 SNMP Service 服务，以使刚刚所作的配置生效。

完成 SNMP 服务配置，并重启了 SNMP 服务后，可以使用 2.3.3 节所介绍的方法对 SNMP 服务是否正常进行验证，这里就不再一一复述了。

2.3.6 通过 SNMP 协议采集监控数据

前面介绍过，其实不管是使用 SNMP 协议的哪个版本采集监控数据，其工作流程和原理在本质上都是相同的。都是 Zabbix 服务器端组件周期性地通过 SNMP 协议，向被监控主机上的 SNMP 服务或其代理发送 SNMP 查询，以获取监控项目所需的监控数据。这种监控数据采集方法原理很简单，Zabbix 服务器端组件（或其代理端组件）根据用户针对不同监控项目所配置的 OID 值，定期地调用 net-snmp 软件包（或 ucd-snmp 软件包）所提供的接口函数，查询被监控主机（或其代理）上 MIB 库中指定 OID 所对应对象的数据，从而采集到被监控项目的监控数据。因此，通过 SNMP 协议（SNMP 陷入除外）采集监控数据的方法是一种被动监控数据采集方法。而使用这种方法采集监控数据，在配置和日常管理上比较简单，只需在被监控主机上开启并配置好 SNMP 服务，并针对 Zabbix 服务器（或其代理服务器）开放查询权限，然后在配置监控项目时指定好监控项目的 OID 值，Zabbix 系统就会按照我们的要求，定时去被监控主机上采集所需要的监控数据，不需要额外地编写数据采集程序或脚本。

或许你会说，我怎么知道我所需要监控的项目的 OID 值，并且如果监控的项目数量很多，那么在 Zabbix 系统中配置监控项目时，需要针对不同的监控项目配置不同的 OID 值，这样我们日常管理工作量岂不是非常大？没错，要想知道某个被监控项目具体的 OID 值，无非有几种方法。第一种方法是，由设备的供应商给我们提供相关的技术文档，从这些技术文档中，可以查出我们所要监控的项目在 MIB 库中的 OID 值。第二种方法是，国际标准化组织已经定义好了通用的 MIB 库结构，所以，一些通用项目，例如系统内存大小、CPU 负载信息、设备网卡数量等项目其 OID 值都是固定的。对于这类监控项目的 OID 值，我们有很多种方法可以得到，比如查阅相关的标准文件，通过互联网搜索，等等。第三种方法是，其实当需要通过 SNMP 协议采集某些监控项目的监控数据时，也可以不需要知道这些监控项目的具体 OID 值。Zabbix 系统为我们提供了一种动态索引方法，用这种方法可以实现自动地发现被监控主机上某类被监控对象的数量，以及它们所对应的 OID 值，并依据所发现的信息，自动地在系统中添加对应的监控项目并实施监控数据的采集。在 Zabbix 系统中，这种自动发现被监控设备上拥有某类对象信息，并自动将其添加为监控项目的功能被称之为低级自动发现功能。关于 Zabbix 系统中的低级自动发现功能，后续章节中有详细的介绍和说明，并会给出相应的实例。

虽然 SNMP 陷入与 SNMP 查询一样，都是 Zabbix 服务器（或其代理）端组件通过 SNMP 协议与被监控主机（或者其代理）通信，以完成监控数据的采集。但是，通过 SNMP 陷入方法采集监控数据与通过 SNMP 协议查询的方法采集监控数据，在工作流程和监控项目的配置上还是有很大的差别的。首先，通过 SNMP 协议查询的方法采集监控数据，是由 Zabbix 服务器端根据监控项目的配置，周期性地连接到被监控主机或其代理的 SNMP 服务上，然后查询监控项目所需要的数据，这属于一种被动的监控数据采集模式。而通过 SNMP 陷入方法采集监控数据则不同，它是当被监控主机上发生某种事件时，由其（或者被监控主机代理）主动地将所发生的事件信息发送给 Zabbix 服务器（或其代理）端组件，由 Zabbix 系统进行分析并获取监控数据，它是一种主动的监控数据采集模式。另外，通过 SNMP 协议查询方式采集监控数据所对应的监

控项目，其关键字（何为监控项目的关键字，我们将在后续章节中作详细的介绍）可以由我们自由定义，只要我们所定义的监控项目关键字符合 Zabbix 系统对监控项目关键字的要求和规则即可。但是，通过 SNMP 陷入方式采集数据所对应的监控项目则不同，它们的关键字具有固定的形式，这个形式是由 Zabbix 系统内部预定义好的，我们只能使用这种形式的关键字，最多只能修改关键字的参数。表 2-3 即为通过 SNMP 陷入方式采集数据所对应监控项目可以使用的关键字列表。

表 2-3 通过SNMP陷入方式采集数据所对应的监控项目关键字列表

关键字	描述	备注
snmptrap[regex]	该关键字的含义是从相应监控接口所接收到的陷入信息中匹配出SNMP陷入信息。其中，regex为所需要匹配的字符串。当Zabbix系统从所接收到的陷入信息中，匹配到这个参数所指定的字符串时，则表示系统接收到了指定监控项目的陷入数据	从前面的描述，即通过SNMP陷入方式采集监控数据的流程中可以看出，当Zabbix服务器上所运行的snmpd服务，接收到了来自被监控主机（或者是其代理）所发送的陷入信息后，会将SNMP陷入信息发给SNMPD进程，由其进行格式化，以便Zabbix服务器端组件能够识别出这条陷入信息。格式化后的陷入信息存放在/etc/snmp/snmpd.conf配置文件中log_file配置项所指定的日志文件里，Zabbix服务器端组件会定时去读取这个日志文件，然后根据每条陷入信息的内容，用监控项目关键字中regex参数所指定的内容去匹配，如果匹配成功了，则表示对应的监控项目接收到了来自被监控主机的一条陷入信息
snmptrap.fallback	这个关键字所对应的监控项目可能有一点不太好理解。针对前一个关键字，我们已经说过，Zabbix服务器会定时地用regex参数所指定的内容去逐条匹配所接收到的SNMP陷入信息，当匹配成功了，就将对应监控项目的监控数据更新为所接收到的陷入信息中的内容。当某台被监控主机上，所有SNMP陷入类监控项目所指定的关键字都没有匹配上所接收到的陷入信息时，而且这台被监控主机上配置了关键字为snmptrap.fallback的监控项目，这个时候，系统就会将这条陷入信息的内容作为监控数据更新到snmptrap.fallback为关键字的监控项目上	需要注意的是，这里所说的系统接收到的陷入信息没有被匹配上，是指该陷入信息在Zabbix系统没有配置相对应的监控项目。但是，在/etc/snmp/snmpd.conf配置文件中，还是得有这类陷入信息格式化配置的。否则，Zabbix系统将会因为无法识别出所接收到的陷入信息，而报告接收到了非法的陷入信息。因此，也就不会更新关键字为snmptrap.fallback所对应监控项目的监控数据了。对于某些我们未知的陷入信息，也可以通过在snmpd.conf配置文件里添加上： EVENT general.* "General event" Normal FORMAT ZBXTRAP \$aA Unknow trap Message 内容，以便对这类陷入信息进行格式化，从而让Zabbix系统能够识别和处理。在snmpd.conf文件里添加上上面这两行内容后，系统会将所有未知的陷入信息内容作为监控数据统一更新到关键字为snmptrap.fallback的监控项目上

2.4 Zabbix 系统内部数据采集

从本质上来说，Zabbix 系统也与其他被监控的系统一样，也是一种应用系统。所以，它的状态、性能数据也需要被采集与监控，以便对它的运行性能、状态情况进行了解，进而在必要的时候对它进行性能调优或进行必要的系统硬件升级，甚至对它的架构进行调整。例如，一般组织，随着组织业务的不断拓展以及实际需要的发展，被监控设备和监控项目的数量也会快速增长。而这种增长达到一定的程度后，原来所采用的单台独立服务器模式的 Zabbix 系统架构就可能很难满足组织的需要了，这个时候，我们可能就需要及时地将 Zabbix 系统由单台独立服务器运行模式调整为分布式运行模式。

而实际上，当决定是否需要对现有的 Zabbix 系统进行硬件升级、系统性能优化甚至更换独立服务器模式为分布式服务器模式时，就要面临依据的问题，即在我们决定执行上述升级前，得有数据依据。

而 Zabbix 系统内部数据的采集，就是采集和监控 Zabbix 系统自身状态和性能数据。Zabbix 系统内部数据是由 Zabbix 服务器端组件通过计算获取的。所以，这种数据采集方法，不需要安装任何客户端。表 2-4 列出的是 Zabbix 系统内部数据采集方法所支持的监控项目关键字列表。

表 2-4 Zabbix系统内部数据采集方法所支持的监控项目关键字列表

关键字(Key)	描述	备注
zabbix[boottime]	该关键字所对应的监控项目，采集 Zabbix 服务器端进程启动的时间戳	时间戳是自 1970 年 01 月 01 日零时的秒数。因此，如果需要转换成年月日的形式，则需要手动转换
zabbix[history]	该关键字所对应的监控项目，采集 Zabbix 系统数据库中 HISTORY 表中记录的条数据	如果 Zabbix 系统数据库使用的是 MySQL InnoDB 存储引擎或 Oracle、PostgreSQL 等数据库，尽量不要使用这个监控项，以免对数据库的性能有较大的影响
zabbix[history_log]	该关键字所对应的监控项目，采集 Zabbix 系统数据库中 HISTORY_LOG 表中记录的条数据	如果 Zabbix 系统数据库使用的是 MySQL InnoDB 存储引擎或 Oracle、PostgreSQL 等数据库，尽量不要使用这个监控项，以免对数据库的性能有较大的影响
zabbix[history_str]	该关键字所对应的监控项目，采集 Zabbix 系统数据库中 HISTORY_STR 表中记录的条数据	如果 Zabbix 系统数据库使用的是 MySQL InnoDB 存储引擎或 Oracle、PostgreSQL 等数据库，尽量不要使用这个监控项，以免对数据库的性能有较大的影响
zabbix[history_text]	该关键字所对应的监控项目，采集 Zabbix 系统数据库中 HISTORY_TEXT 表中记录的条数据	如果 Zabbix 系统数据库使用的是 MySQL InnoDB 存储引擎或 Oracle、PostgreSQL 等数据库，尽量不要使用这个监控项，以免对数据库的性能有较大的影响
zabbix[history_uint]	该关键字所对应的监控项目，采集 Zabbix 系统数据库中 HISTORY_UINT 表中记录的条数据	如果 Zabbix 系统数据库使用的是 MySQL InnoDB 存储引擎或 Oracle、PostgreSQL 等数据库，尽量不要使用这个监控项，以免对数据库的性能有较大的影响

续表

关键字(Key)	描述	备注
zabbix[host,<type>,available]	该关键字所对应的监控项目，采集指定的主机是否支持某种类型的数据采集方法。这类监控项目所采集的值与主机列表（关于如何查看系统中主机列表，将在后续章节中有详细介绍）中的数据采样方法可用性图标状态是相对应的	该关键字中的type参数可选值有agent、snmp、ipmi及jmx等。如果返回值为0,则表示该主机对应的数据采集方法不用；如果返回值是1，则表示该采集方法可用；而如果返回值为2，则表示对应的数据采集方法是否可用为未知
zabbix[items]	该关键字所对应的监控项目，采集系统中已配置的监控项目总数	
zabbix[items_unsupported]	该关键字所对应的监控项目，采集系统中已配置的，但是不被支持的监控项目总数	
zabbix[java,<param>]	该关键字所对应的监控项目，采集与Zabbix Java应用程序网关相关的信息	如果<param>参数取ping的话，那么当Java应用程序网关可用时，则返回数字1。针对该关键字所对应的监控项目，可以在触发器中使用nodata()函数，以识别出Java应用程序网关可用状态，并在必要的时候报警。当<param>参数取值为version时，则返回Java应用程序网关的版本信息，例如2.0.0等
Zabbix[process,<type>,<mode>,<state>]	该关键字所对应的监控项目，采集的是指定的一个Zabbix进程或指定的一组Zabbix进程（通过<type>和<mode>两个参数来指定）处在<state>参数指定的状态下的时间占总的时间的百分比。该数据是以系统的最后一分钟数据作为依据进行计算的	Zabbix服务器端进程有多达20多种（当在Linux操作系统下用ps -ef命令来查看时，往往可以看到很多Zabbix系统进程。而这些系统进程在Zabbix内部称为实例。这些实例各自负责不同的工作，就形成了不同种类型的进程）。在该类监控项目中，<mode>参数可以被指定为某种类型进程的进程号。需要注意的是，这里所谓的进程号，并不是系统中的进程号，而是Aabbix系统中某种类型进程的序号号。例如，如果在Zabbix服务器组件的配置文件中用StartPollers参数指定启动5个轮询进程，那么mode参数可以指定的范围为1~5。而如果所指定的数据超过这个范围，则所创建的监控项目就将会报不被支持的错误。 该关键字参数说明见本表后【1】所述
zabbix[proxy,<name>,<param>]	该关键字所对应的监控项目，采集与Zabbix服务器代理相关的数据信息	参数<name>用于指定服务器代理名称。可选<param>参数值有lastaccess，表示最近一次接收来自服务器代理信息的时间戳。针对该关键字所对应的监控项目，触发器函数fuzzytime()可用于检查服务器代理的可用性

续表

关键字(Key)	描述	备注
zabbix[proxy_history]	该关键字所对应的监控项目，采集 Zabbix 服务器代理端 proxy history 表中等待发送给 Zabbix 服务器端数据的条数	
zabbix[queue,<from>,<to>]	该关键字所对应的监控项目，采集系统队列中延时了<from>到<to>时长尚未采集到最近一次监控数据的监控项目的个数	from——默认值是6秒 to——默认值是无穷大
zabbix[rcache,<cache>,<mode>]	该关键字所对应的监控项目，采集内存中用于缓存配置信息的内存空间使用信息	参数cache可选值为buffer 参数mode可选的值有： total——返回缓存配置信息所占用的总的内存空间大小。这个返回值应与Zabbix服务器端组件的配置文件中CacheSize配置选项指定的数值是一致的 free——返回用于缓存配置信息的空闲的内存空间大小 pfree——返回用于缓存配置信息的空闲的内存空间占总的内存空间的百分比 used——返回用于缓存配置信息的已使用的内存空间大小
zabbix[requiredperformance]	该关键字所对应的监控项目，采集的是 Zabbix 服务器期望的性能值，即 Zabbix 系统根据系统中所配置的被监控主机和监控项目的数量及类型，估算出来的服务器每ns需要处理的新数据的数量	该关键字所对应的监控项目的返回值与我们将在后面将要介绍的“状态统计”→“状态面板”→“ZABBIX状态”及“系统报告”→“ZABBIX状态”中的“服务器性能（值/秒）”项目显示的数值是相同的，且其含义也是一样的
zabbix[trends]	该关键字所对应的监控项目，采集 Zabbix 系统数据库中 trends 表中记录的条数据	如果 Zabbix 系统数据库使用的是 MySQL InnoDB 存储引擎或 Oracle、PostgreSQL 等数据库，尽量不要使用这个监控项，以免对数据库的性能有较大的影响
zabbix[trends_uint]	该关键字所对应的监控项目，采集 Zabbix 系统数据库中 trends_uint 表中记录的条数据	如果 Zabbix 系统数据库使用的是 MySQL InnoDB 存储引擎或 Oracle、PostgreSQL 等数据库，尽量不要使用这个监控项，以免对数据库的性能有较大的影响
zabbix[triggers]	该关键字所对应的监控项目，采集 Zabbix 系统中已经配置的触发器数量	
Zabbix[wcache,<cache>,<mode>]	该关键字所对应的监控项目，用于采集 Zabbix 系统中写缓存信息	该关键字参数说明见本表后【2】所述

【1】Zabbix[process,<type>,<mode>,<state>]关键字参数说明

该关键字所对应的监控项目支持的进程类型（由 type 参数指定）包括：

- ❑ 报警器进程(alerter)。该类型的进程是用来发送报警通知的。
- ❑ 配置同步器进程(configuration syncer)。该类型的进程用于将配置文件中的配置信息同步到内存缓存中。
- ❑ 数据库看门狗进程(db watchdog)。该类型进程用于监视 Zabbix 系统数据库状态。当数据库状态变为不可用时，它将发送警告信息（服务器代理端不支持该类型的进程）。
- ❑ 自动发现器进程(discoverer)。该类型进程用于自动发现设备。
- ❑ 步骤进程(escalator)。该类型进程是用于处理动作中步骤升级的。
- ❑ 历史数据同步器进程(history syncer)。该类型进程是用于写历史数据表的。
- ❑ 管家进程(housekeeper)。该类型进程是用于清理过期的历史数据的。
- ❑ HTTP 轮询器进程(http poller)。该类型进程是用于轮询 Web 类监控项目的。
- ❑ ping 检查器进程 icmp pinger)。该类型进程是用于定期进行 ICMP PING 检查的。
- ❑ ipmi 轮询器进程 (ipmi poller)。该类型进程是用于定期进行 ipmi 类监控项目检查的。
- ❑ java 轮询器进程(java poller)。该类型进程是用于轮询 java 类监控项目的。
- ❑ 分布式节点看守器进程(node watcher)。该类型进程是用于在不同的分布式节点发送历史数据和配置信息更新的。
- ❑ 轮询器进程(poller)。该类型进程是用于普通的被动监控项目的轮询的。
- ❑ 服务器代理轮询进程(proxy poller)。该类型进程是用于服务器代理被动轮询的。
- ❑ 自我监控进程(self-monitoring)。该类型进程是用于收集 Zabbix 系统内部监控信息的。
- ❑ 定时器进程(timer)。用于处理触发器中与时间相关的函数和维护模式的进程。
- ❑ 陷入器进程(trapper)。该类型进程是用于处理主动采集、陷入及分布式节点间或服务代理的。
- ❑ 不可到达轮询器进程(unreachable poller)。该类型进程是用于轮询不可到达的设备的。
- ❑ vmware 收集器进程(vmware collector)。该类型进程是用于负责从 vmware 服务进程中收集数据（Zabbix 服务器代理端不支持这种类型的进程）的。

可选的 mode 参数值包括：

- ❑ avg。平均值。
- ❑ count。返回创建的指定类型进程的数量。
- ❑ max。最大值。
- ❑ min。最小值。
- ❑ <process number>。进程号，含义参见“描述”中所述。

可选的 state 参数值包括：

- ❑ 繁忙(busy)。繁忙状态。
- ❑ 空闲(idle)。空闲状态。

【2】Zabbix[wcache,<cache>,<mode>]关键字参数说明

参数 cache 可以选择 values、history、trend、text 等值。当参数 cache 选择 values 值时，则

mode 参数可以选择以下值:

- **all**. 返回 Zabbix 服务端进程或服务器代理端进程自启动以来处理的所有监控数据的总数。但是不包括不被支持的监控项目。
- **float**. 返回 Zabbix 服务端进程或服务器代理端进程自启动以来处理的所有浮点型监控数据的总数。
- **uint**. 返回 Zabbix 服务端进程或服务器代理端进程自启动以来处理的所有无符号整型监控数据的总数。
- **str**. 返回 Zabbix 服务端进程或服务器代理端进程自启动以来处理的所有字符串型监控数据的总数。
- **log**. 返回 Zabbix 服务端进程或服务器代理端进程自启动以来处理的所有日志型监控数据的总数。
- **text**. 返回 Zabbix 服务端进程或服务器代理端进程自启动以来处理的所有文本型监控数据的总数。
- **not supported**. 返回 Zabbix 服务端进程或服务器代理端进程自启动以来处理的不被支持的监控数据的总数。

当参数 cache 选择 history 值时, 则 mode 参数可以选择以下值:

- **pfree**. 返回的是历史缓冲区中, 空闲缓冲区占总的历史缓冲区的百分比。总的历史缓冲区的大小是由 Zabbix 服务器或其代理端组件的配置文件中的 HistoryCacheSize 配置项所指定的。历史缓冲区是用来缓存所有类型监控项目信息、采集数据的时间戳以及数值类型监控项目所采集的数据的。如果该监控项目所采集的数据值偏低, 则很可能意味着 Zabbix 系统数据库方面出现了性能问题。
- **free**. 返回空闲的历史缓冲区大小。
- **total**. 返回总的历史缓冲区大小。
- **used**. 返回已使用的历史缓冲区大小。

当参数 cache 选择 trend 值时, 则 mode 参数可以选择以下值:

- **pfree**. 返回的是趋势缓冲区中空闲缓冲区占总的趋势缓冲区的百分比。总的趋势缓冲区的大小是由 Zabbix 服务器或其代理端组件的配置文件中的 TrendCacheSize 配置项所指定的。趋势缓冲区用于缓存监控项目在最近一个小时内所采集到的所有数据, 以用于数据聚合使用。服务器代理端不支持该种数据采集方法。趋势缓冲区其他统计项目, 如 free、total、used 等, 它们与历史缓冲区中对应的统计项目的含义类似, 在这里我们不再一一赘述。

当参数 cache 选择 text 值时, 则 mode 参数可以选择以下值:

- **pfree**. 返回的是文本缓冲区中空闲缓冲区占总的文本缓冲区的百分比。总的文本缓冲区的大小是由 Zabbix 服务器或其代理端组件的配置文件中的 HistoryTextCacheSize 配置项所指定的。文本缓冲区是用于缓存字符、文本以及日志类型历史数据的。但是, 上述这些类型的监控项目信息以及所采集的监控数据的时间戳信息仍然是保存在历史缓冲区中的。文本缓冲区其他统计项目还有 free、total、used 等, 它们与历史缓冲区中对应统计项目的含义类似, 在这里我们不再一一赘述。

2.5 Zabbix 陷入

前面在介绍通过 SNMP 协议采集监控数据的方法时介绍过 SNMP 陷入，因此，相信大家对陷入应该有一定的了解。与 SNMP 陷入类似，Zabbix 陷入是指在被监控主机上定时地执行特定的程序或命令，主动向 Zabbix 服务器或其代理服务器端发送监控数据的监控数据采集方法。因此，Zabbix 陷入是一种主动模式的数据采集方法，即由被监控主机定时和周期地向 Zabbix 系统的服务器端或服务器代理端发送所采集到的监控数据。与 SNMP 陷入需要 SNMP 协议支持不同，Zabbix 陷入只需使用命令行工具（在编译安装 Zabbix 系统时，如果选择编译安装了 zabbix_agent 模块，则系统默认会编译安装 zabbix_sender 命令行工具。通过这个命令行工具，就可以向 Zabbix 服务器端或服务器代理端发送 Zabbix 陷入信息）。或使用以任何一种编程语言编写的脚本或程序即可向 Zabbix 系统的服务器端或服务器代理端发送 Zabbix 陷入信息。

使用 Zabbix 系统自带的工具 zabbix_sender 发送 Zabbix 陷入信息的方法很简单，只需周期性地执行如下命令即可：

```
shell> ./zabbix_sender -z <Zabbix 服务器 IP 地址或主机名> -p <Zabbix 服务器端或服务器代理端的端口号 (默认是 10051)> -s <被监控主机的主机名> -k <监控项目的关键字> -o <所要发送的数据>。
```

使用程序或脚本发送 Zabbix 陷入信息也比较简单，只需使用任何一种编程语言编写脚本或程序，定期地向 Zabbix 服务器或服务器代理端指定端口发送符合 Zabbix 系统接口规范的数据信息即可。具体如何使用编程的方法来发 Zabbix 陷入信息，我们将在后续章节作详细阐述。

但是，在使用 Zabbix 陷入的方法采集监控数据时需要注意以下两点：

- 当通过 Zabbix 系统的 Web 前端页面配置好一个 Zabbix 陷入类型的监控项目后，需要等待一定的时间后，Zabbix 服务器或其代理才能接收来自被监控主机所发来的陷入信息。原因是因为，Zabbix 服务器端相关进程会定期地将系统中的配置信息更新到配置信息缓冲区中，而这个时间的长短则由服务器端配置文件中的 CacheUpdateFrequency 配置项来决定，默认是 60 秒。
- zabbix_sender 命令行工具支持从指定的文件中读取需要发送的监控项目信息及所采集的数据，其方法是在执行 zabbix_sender 命令时使用 -i 参数。该命令行更详细的使用方法可查看其帮助文档，我们在这里不再一一说明。

2.6 数据聚合

所谓数据聚合（在 Zabbix 系统前台页面上的监控项目类型中称之为 Aggregate Checks），是指并不直接从被监控主机上采集监控数据，而是根据已定义的监控项目上所采集的监控数据，通过一定的计算方法计算出新的监控项目所需的监控数据。例如，如果想统计某个二层交换机的上联口的出口流量，因为交换机是二层的，所以我们很难直接对交换机的端口进行数据采集和监控。这个时候，我们只要将连接到这台交换机上所有主机的对应网卡的出口流量进行聚合，就可以大致上得出这个二层交换机的上联口的出口流量（严格地说，如果与这台交换机相连的主机之间有通过这台交换机交换的数据，那么这部分流量需要减去）。再比如，在网络游戏行业，我们经常需要监控游戏的在线人数。而当需要统计某个区组里所有游戏服务器的在线人数时，只需将属于这个区组里的所有游戏服务器的在线人数数据进行聚合就可以了。

使用数据聚合不需要在 Zabbix 服务器、Zabbix 代理服务器和被监控主机上额外安装任何插件和工具。数据聚合是 Zabbix 服务器上运行的相关进程，它直接通过查询 Zabbix 系统自己的数据库中相关的数据，并经过一定的计算来完成聚合。所以，从这一点上来说，数据聚合为我们提供了一种对现有的数据进行再挖掘和利用的好方法。

当你读到后面 2.13 节将要介绍到的“通过计算的方法采集监控数据”这一节的时候，你可能会有一些迷惑。数据聚合和通过计算的方法采集监控数据，在本质上都是在现有的监控数据的基础上，通过一定的方法进行计算从而获得新的监控数据的方法。你思考的没有错，是这样的，在本质上这两种数据采集方法是很类似。但是，这两种数据采集方法也有一些差别。数据聚合主要针对的是不同主机层面的监控数据的计算，它可以先对被聚合的监控项目所采集到的数据求平均值、求最大值等，然后再将一组或多组这样的平均值、最大值基于不同的主机求最大值、平均值等。而通过计算的方法采集监控数据却做不到这一点，它只能对同一个主机的不同监控项目或不同主机上的多个监控项目所采集的监控数据进行计算。计算完成之后，将计算所得到的数据作为新监控项目所采集到的监控数据存入数据库中，不能像数据聚合那样再次依据主机组进行计算。另外，通过计算的方法采集监控数据，如果被计算的监控项目是跨多个主机的，则其计算方法上也没有数据聚合那样灵活和方便。如前所述，数据聚合方法提供了一些数据聚合函数，表 2-5 及表 2-6 所列的函数便是可以直接应用到主机层面和监控项目层面的聚合函数。对于这些函数，读者目前无须过多地纠结于如何使用它们，而只需了解 Zabbix 系统的数据聚合方法是有其支持的函数的，至于如何使用，我们在后续章节中会有陆续说明。

表 2-5 数据聚合方法中组支持的函数列表

序号	函数名	函数功能
1	grpavg	求平均值
2	grpmax	求最大值
3	grpmin	求最小值
4	grpsum	求和

表 2-6 数据聚合方法中监控项目支持的函数列表

序号	函数名	函数功能
1	avg	求平均值
2	count	求采集到的数据个数
3	last	求采集到的最后一个值
4	max	求最大值
5	min	求最小值
6	sum	对指定时间段内采集到的数据求和

读者可能会有一些迷惑，又是组函数，又是监控项目函数的。下面我们仍以上面提到的例子为例来加以说明。当要聚合一组主机所有外网网卡的出口流量，并将其作为一个新的监控项目的监控数据时，那么，就只需将这组主机中每台主机指定网卡的出口流量的最后一个值都取出来，然后将这取出的一组数据进行求和，就得出了最近一个时刻的交换机的上联口的出口流量。这个时候，在监控项目上就需要使用 last 函数，而在组上就需要使用求和函数。再比如，如果想知道上述这组主机中，某一段时间内某台主机所占用的最大流量值是多少，就只需首先

对每台主机上的网卡流量进行求和，然后再在这组求和的值中找出最大的值。这个时候，就需要首先对监控项目使用求和函数，即 `sum` 函数，然后再对所得到的这组数据求最大值。

2.7 通过脚本采集监控数据

通过脚本采集数据，在 Zabbix 系统中称之为外部检查（External checks）。在本书中，我们从这种数据采集方法的本质出发，将其称之为通过脚本采集数据。这种数据采集方法其实很简单，即由 Zabbix 服务器端进程根据配置信息，按计划地执行指定的脚本程序或命令，然后俘获脚本程序或命令的输出内容作为对应监控项目所采集到的监控数据。这个被执行的脚本程序或命令的存放路径由 Zabbix 服务器端组件的配置文件中 `ExternalScripts` 配置项所指定。默认这个配置项的值为 Zabbix 安装路径下的 `share/zabbix/externalscripts` 目录。读者可以修改这个配置项指定的路径，但是需要注意的是，对应的路径对于运行 Zabbix 服务器端组件进程的用户（默认是 Zabbix 用户）至少需要有可读和可执行权限。同时，对于具体被 Zabbix 服务器端进程或服务器代理端进程执行的脚本程序或命令，运行 Zabbix 服务器端组件进程的用户至少也需要有可读和可执行权限。否则，对应的脚本程序或命令将无法被执行。

需要注意的是，Zabbix 系统将以捕获的脚本程序或命令的输出内容作为对被监控项目所采集到的监控数据，它无法通过脚本程序或命令的退出码来判断脚本程序或命令是否被正确执行。换句话说，即便被执行的脚本程序或命令出错，Zabbix 系统也无法识别出来，而只是简单地将错误信息（如果有错误信息输出的话）作为监控项目所采集到的监控数据来看待。

另外，通过脚本的方法采集监控数据，对 Zabbix 系统的性能影响比较大。所以，一般情况下，我们应尽量少使用这种方法来采集监控数据。但是，通过脚本采集监控数据的方法有一个很大的优势，那就是它不需要在被监控主机上安装任何的工具或软件，而且，采集数据的脚本程序也可以使用任何一种编程语言来编写。

使用脚本程序采集监控数据的监控项目配置很简单，只需在创建监控项目时，选择 `External check` 监控项目类型，然后在监控项目的关键字（key）栏里输入如下格式的关键字即可：

脚本/命令文件名[<参数 1>,<参数 2>,...]

例如，`echo_data.sh[{HOST.HOST}]`。

2.8 数据库监控

数据库监控(Database Monitoring)在 Zabbix 系统的 Web 前端组件中被称之为数据库监控，而在其服务器端组件里被称之为 ODBC 监控(ODBC Monitoring)。在第 1 章介绍 Zabbix 系统安装时，我们简单地对数据库监控做过一些介绍。

开放数据库互连(Open Database Connectivity, ODBC)是微软提出的一种数据库访问接口标准。它是微软公司开放服务结构(Windows Open Services Architecture, WOSA)中有关数据库的一个重要组成部分。它建立了一组规范，并提供了对数据库访问的标准应用程序接口(API, Application Programming Interface)。通俗地说或者从应用者的角度来说，ODBC 就是一个提供了访问 SQL 类型数据库标准应用程序接口的中间件，即通过 ODBC 提供的标准 SQL 接口，可以访问众多类型的关系型数据库，例如，MySQL、SQL Server、Oracle、DB2，等等，而不需要从应用程序层面过多地关心后端到底运行的是哪种类型的关系型数据库。具体到应用程序（我

们在这里也可以将 Zabbix 系统服务器端进程简单地视为一种应用程序) 通过 ODBC 查询数据库的过程是: 应用程序通过 ODBC 中间件所提供的接口函数, 来调用某个特定数据库的驱动程序接口, 从而执行数据库查询, 然后数据库驱动程序将所查询到的结果逐级返回给应用程序。

数据库监控采集监控数据方法的原理就是, Zabbix 服务器相关进程直接通过 ODBC 接口技术, 查询各种不同类型的关系型数据库里的数据, 以采集需要的数据。由此可以看出, 数据库监控是一种通过直接操作被监控系统数据库的方法来采集监控数据的, 这在某些场景下非常高效和有用。例如, 监控论坛的在线人数情况, 只需通过数据库监控技术, 直接读取论坛数据库中记录在线人数的数据表中相关信息即可。另一方面, 通过数据库监控的方法采集监控数据, 我们不需要在被监控主机上安装任何额外的软件, 只需将对应数据库的权限对 ODBC 数据源开放即可。

因为是通过 ODBC 接口技术连接被查询数据库的, 所以使用数据库监控方法可以采集任何支持 ODBC 连接的关系型数据库中的数据。目前, Zabbix 系统支持 unixODBC 和 iODBC 两种 ODBC 中间件。所以, 要想让 Zabbix 系统能通过这种方法采集监控数据, 则需要在 Zabbix 服务器端安装这两个中间件中的任何一个, 且在编译安装 Zabbix 服务器端组件时需要显式地开启 ODBC 支持。即在编译配置时需要带上 `--with-unixodbc` 或 `--with-iodbc` 配置选项。在本书中, 选择安装 unixODBC。

在使用数据库监控的方法采集监控数据时需要注意以下几点:

- ❑ 执行 SQL 查询的最长时间不得超过 Zabbix 服务器端组件配置文件中 Timeout 选项指定的超时时间, 否则将导致查询失败。
- ❑ 每次每个监控项目所对应的数据库查询只能返回一个值。
- ❑ 如果查询的返回值多于一列的话, 则只有第一列的数值才有效。
- ❑ 如果查询的返回值多于一行的话, 则只有第一行的值才有效。综合上述的注意事项, 不难看出, Zabbix 系统只取查询结果中的第一行、第一列的数值。
- ❑ 每条 SQL 查询语句必须以 SELECT 子句开头。
- ❑ 每条 SQL 查询语句必须写在一行, 中间不得有换行或回车。

2.9 通过 IPMI 代理采集监控数据

智能平台管理接口 (Intelligent Platform Management Interface, IPMI), 是管理基于 Intel 架构的企业系统外围设备所使用的一种工业标准。该标准由英特尔、惠普、NEC、戴尔和 SuperMicro 等公司所制定。通过 IPMI, 可以很方便地采集和监控到服务器或其他类型主机的硬件状态, 例如 CPU 风扇的转数、CPU 温度、电压等。实际上, 通过 IPMI 接口, 我们不但可以采集到主机的硬件状态, 而且通过专门的软件我们还可以实现远程开关机 (只要主机已经接好电源, 且 IPMI 连接已配置好), 以及远程查看远程开机到操作系统完全启动之前的过程信息。所以, IPMI 的最大优势在于, 它独立于主机的 CPU 和操作系统, 无论主机是在开机还是关机的状态下, 只要主机已经接好了电源并且配置好了 IPMI 的链路, 用户就可以随时远程对这台主机进行操控。这个特性在某些时候对于运维来说是非常有用的。比如说服务器的根文件系统损坏, 这个时候, 如果没有 IPMI 技术, 我们可能必须要赶到现场, 接上显示器和鼠标、键盘才能执行对根文件系统的修复操作。

IPMI 的核心部件是一种叫做主板管理控制器 (Baseboard Management Controller, BMC)

的嵌入式管理微控制器。它不依赖于主机的处理器、BIOS 和操作系统而独立工作。所以,通过 IPMI 可以做到在关机状态进行远程开机等操作。也正是因为这个原因,当通过下面介绍的方法给 IPMI 配置了 IP 地址、网关等信息,但是在 IPMI 所对应的网络接口不接通网线的话,即使是在同一台主机上也 ping 不通 IPMI 模块上配置的 IP 地址(在一台主机上,如果有多块网卡,只要我们分别配置上 IP 地址,即使不连接网线,只要网卡工作正常,那么在这台主机上就可以 ping 通所有网卡上所配置的 IP 地址)。因此,我们在调试 IPMI 模块的网络连通性时需要注意这一点,不能将其视为跟操作系统可以管理的其他网络接口一样,而应该将其视为另一台逻辑上独立的主机。实际上,很多时候主机上的 IPMI 模块就是一块插在主板上的板卡,当然,现在很多产品已经将其集成到主板上了。虽然使用 IPMI 有很多优点,但是并不是所有的主机都支持 IPMI。所以,在确定使用 IPMI 方法采集监控数据之前,应确定对应设备是否支持通过 IPMI 来管理。

IPMI 现在一般可以支持串口和网络连接。既然支持网络连接,那么就必须有网络接口。IPMI 网络接口在不同品牌的设备上叫法可能不太一样,例如 HP 称之为 iLO,而 DELL 称之为 DRAC。

要使 Zabbix 系统能够通过 IPMI 方法采集监控数据,则被监控设备上的 IPMI 也要做一些配置。下面介绍如果配置被监控设备上的 IPMI 模块。这里以 CentOS 操作系统为例。

首先,要在被监控主机上安装 OpenIPMI 和 OpenIPMI-tools 软件包,这里我们通过 yum 方法来安装。

```
shell> yum install OpenIPMI
shell> yum install OpenIPMI-tools
```

安装完成之后,通过下列命令启动 IPMI 服务:

```
shell> /etc/init.d/ipmi start
```

如果安装和启动 IPMI 服务都没有报错,则说明 IPMI 软件包安装正确。也可以使用下列命令检查 IPMI 工作是否正常:

```
shell> ipmitool -I open sensor list |more
```

如果有如图 2-11 所示的输出,则说明 IPMI 服务工作正常。接下来就需要配置 IPMI 网络和用户参数了。

```
shell> ipmitool lan set 1 ipaddr 192.168.5.120      # 配置网络 IP 地址
shell> ipmitool lan set 1 netmask 255.255.255.0    # 配置网络子网掩码
```

temp	na	degrees C	na	na	na	na	na	85,000	90,000	na
temp	na	degrees C	na	na	na	na	na	85,000	90,000	na
temp	na	degrees C	na	na	na	na	na	na	na	na
temp	na	degrees C	na	na	na	na	na	na	na	na
ambient Temp	22,000	degrees C	ok	na	3,000	3,000	na	42,000	47,000	na
fan fan Temp	na	degrees C	na	na	3,000	3,000	na	92,000	97,000	na

图 2-11 IPMI 输出信息

```
shell> ipmitool lan set 1 defgw ipaddr 192.168.5.1  # 配置默认网关
shell> ipmitool lan set 1 access on                 # 开启 IPMI Over LAN 功能
#打印网络配置情况,以检查配置是否正确
shell> ipmitool lan print 1
# 配置一个用户 ID 为 12, 用户名为 monitor 的用户
shell> ipmitool user set name 12 monitor
# 设置用户 ID 为 12 的密码为 password
shell> ipmitool user set password 12 password
shell> ipmitool user enable 12                      # 启用用户 ID 为 12 的用户
#设置用户 ID 为 12 的用户的权限为 2 级。数字越大权限越高。最大值为 4 即管理员
shell> ipmitool user priv 12 2 1
```

提示：如果能重启服务器，并且可以在现场修改服务器 BIOS 的设置，则上面这些配置信息，也可以直接通过服务器的 BIOS 来设置。

2.10 通过 SSH 协议采集监控数据

相信实际维护过 Linux 系统的朋友，对于 SSH 协议应该不陌生。安全外壳协议(Secure Shell, SSH)，它是我们日常远程登录到一台 Linux 系统所常用的协议，其是由 IETF 网络工作小组制定的。它相对安全、可靠，支持多种登录认证方式。比较常用的登录认证方式有用户名和密码方式、公钥和私钥认证方式等。

要想让 Zabbix 系统支持通过 SSH 协议采集监控数据，则需要在编译安装 Zabbix 服务器端和其代理端组件时显式地开启 SSH 协议支持，即在编译配置时使用 `--with-ssh2` 配置选项，并安装 `libssh2` 动态库支持。

通过 SSH 协议采集监控数据的原理很简单，Zabbix 服务器端或服务器代理端根据用户的配置，使用 SSH 协议连接到被监控主机上，然后在被监控主机上执行用户在监控项目上所配置的命令序列或者指定的脚本。Zabbix 服务器端或其代理端捕获命令序列或脚本程序被执行后所返回的结果，并将其作为监控项目所采集的监控数据。

前面我们已经提到过，在使用 SSH 协议远程登录某个 *nix 系统时，最常用的认证方式是用户密码方式和公钥私钥方式两种方式。Zabbix 系统也支持以这两种认证方式远程连接到被监控主机上，并在被监控主机上执行指定的命令序列或脚本程序。对于用户密码认证方式，在 Zabbix 系统中配置起来很简单，只需在配置监控项目时配置用于连接目标主机的用户名和密码即可。所以，在这里我们对这种认证方式就不做过多的叙述。

相对于用户名和密码的认证方式，用公钥私钥的认证方式有一个很明显的优势。出于安全方面的考虑，需要经常定期或不定期地修改被监控主机上用户的密码，以防止因为密码长期未做修改而导致泄漏，从而引起系统安全方面的风险。在这种情况下，一旦我们修改了被监控主机上用户的密码，则也需要在 Zabbix 系统中相应地将被监控项目上所配置的用户名密码修改过来，否则将无法采集到数据。如果在 Zabbix 系统中，这类监控项目的数量很多，则定期修改 Zabbix 系统中所配置的用户名密码将是一项很大工作量的任务。同时，使用用户名密码认证方式，还需要将用户名和密码以明文的方式配置在 Zabbix 系统中，这无疑也会给被监控主机的系统安全带来很大的隐患。故此，一般在实际的系统监控中，很少直接使用用户名密码的认证方式来通过 SSH 协议采集监控数据。而使用公钥私钥认证方式则不会有这些问题，一般只需按照要求生成公钥和私钥对文件，并将公钥信息配置在被监控主机指定用户的特定文件里，然后在 Zabbix 系统中配置相应的公钥私钥认证方式，就可以通过 SSH 协议采集监控数据了。即便我们以后修改了被监控主机上用户的密码，只要公钥和私钥信息没有改变，则不需要对 Zabbix 系统中的配置作任何修改和调整。同时，公钥和私钥文件里的信息都是通过一定的加密算法加密的。所以，只要我们保管好私钥信息和文件，一般不会带来很大的安全风险。

在通过 SSH 协议采集数据的方法中，采用公钥私钥认证方式需要对 Zabbix 服务器和被监控主机做一些配置。下面就来详细介绍一下如何配置 SSH 协议的公钥和私钥信息。

首先，在 Zabbix 服务器端（或 Zabbix 服务器代理端）通过下列系统命令创建公钥和私钥文件。需要注意的是，运行下列命令需要以运行 Zabbix 服务器端进程所使用的用户身份进行。

一般运行 Zabbix 服务器端进程的用户为 zabbix。同时，需要对应用户的目录是存在的，且对这个用户是可读写的。

```
shell> su - zabbix # 切换到 zabbix 用户下
shell> ssh-keygen -b 1024 -t rsa # 创建公钥私钥文件
```

执行 ssh-keygen 命令时，系统会给出多个输入提示，一路回车不需要输入任何内容即可。成功执行上述两条命令后，系统会在 /home/zabbix/.ssh/（如果是以 zabbix 用户的身份执行上述命令，且 zabbix 用户的目录为 /home/zabbix 的话，则所生成的公钥私钥文件就存在 /home/zabbix/.ssh 目录下。当然，不同的发行版，系统所生成的公钥和私钥文件存放的路径可能不同）。在该目录下分别创建 id_rsa.pub 和 id_rsa 文件。这两个文件分别对应公钥和私钥文件。接下来，就需要通过如下命令，将所生成的公钥文件复制到所有被监控主机上。

```
shell> ssh-copy-id root@192.168.5.139
```

需要注意的是：一、上述这条命令中 root 即为将来 Zabbix 系统进程连接被监控主机时所使用的用户账号。这个用户账号必须是被监控主机系统中存在的真实用户账号，且需要 shell 支持。二、SSH 服务的默认端口是 22，通过上述这条命令复制公钥文件时，系统将会以默认的 22 端口去连接被监控主机系统。然而，出于安全方面的考虑，我们经常将 SSH 服务的端口号修改为非 22 端口。这样的话，则在执行上述这条命令之前，先需要对 ssh-copy-id 这个脚本程序作一点适当的修改，否则将会出现连接不上的错误。是的，你没有看错，ssh-copy-id 是一个脚本程序，而不是一个二进制系统命令。所以，可以很方便地对它进行修改，我们可按下列方法修改 ssh-copy-id 文件：

```
shell> cp /usr/bin/ssh-copy-id /usr/bin/ssh-copy-id_org # 备份源文件
# 用 vi 打开 ssh-copy-id 脚本文件，并对它进行如下修改：
shell> vi /usr/bin/ssh-copy-id
```

修改该文件中如下行：{ eval "\$GET_ID"; } | ssh \$1 "umask 077....."，为 { eval "\$GET_ID"; } | ssh -p 2228 \$1 "umask 077....."。即在 ssh 命令后面添加上 -p 2228 内容。

当公钥文件复制到被监控主机上后，可以使用下列命令尝试从 Zabbix 服务器上远程连接被监控主机。如果连接成功，则说明在被监控主机上已经配置好了 Zabbix 服务器的公钥：

```
shell> ssh [-p 2228] root@192.168.5.139
```

接下来，还需要修改 Zabbix 服务器和 Zabbix 服务器代理端组件的配置文件 zabbix_server.conf 和 zabbix_proxy.conf，以告诉 Zabbix 系统，当它准备通过公钥私钥方式连接被监控主机时，它应该到哪里去找所需要的公钥和私钥文件，即指定存放公钥和私钥文件的路径。如前面所述，系统生成的公钥和私钥文件一般存放在运行 Zabbix 服务器端进程的系统用户的目录下的 .ssh 目录下。所以，这个路径一般就被指定为 /home/zabbix/.ssh。

```
#用 vi 打开 Zabbix 安装目录下的 etc/zabbix_server.conf 文件或 etc/zabbix_proxy.conf 文件
shell> vi etc/zabbix_server.conf
修改 #SSHKeyLocation= 行为 SSHKeyLocation=/home/zabbix/.ssh。
```

知道了通过 SSH 协议采集监控数据的原理，我们不难理解，在 Zabbix 系统中所配置的被执行命令序列或脚本程序文件，最终是在被监控主机上执行。所以，要确保这些命令序列或脚本程序能够在被监控主机上正确执行，就需要确保被监控主机上存在这些被执行的命令和脚本文件，且 Zabbix 系统连接被监控主机的用户需要具有相应的权限。同时，在被监控主机上运行的命令序列或脚本程序，可以使用 Zabbix 系统连接被监控主机时所使用用户的环境变量。这些命令序列和脚本程序执行的效果，与我们手工使用相同用户登录到对应被监控主机时，执行这些命令序列和脚本程序的效果是一样的。

2.11 通过 TELNET 协议采集监控数据

在前面提到过，如今要远程登录到一台 Linux 服务器，最通常的做法是通过 SSH 协议远程登录到远程服务器上。但是，实际上，SSH 协议大概是在 1995 年才出现的。所以，在 SSH 协议出现之前，通过 TELNET 协议远程登录到远程服务器是最常用的做法。即使到现在，还有很多设备，特别是一些较旧的网络设备，仍然只支持 TELNET 协议远程连接。所以，对于这些设备监控数据的采集，通过 TELNET 协议方法不失为一种较好的方法。

而通过 TELNET 协议采集监控数据，与我们刚刚在前面介绍的通过 SSH 协议采集监控数据是类似的，只是 Zabbix 服务器（或其代理服务器）与被监控主机之间的通信所采用的协议不同而已。由此不难理解，所谓通过 TELNET 协议采集监控数据，就是 Zabbix 服务器端进程（或其代理端进程）通过 TELNET 协议远程连接到被监控主机，并在远程被监控主机上运行指定的命令序列或脚本程序，然后 Zabbix 服务器端进程（或其代理端进程）捕获命令序列或脚本程序执行完成后的返回信息，作为监控项目所采集到的监控数据。但是，与通过 SSH 协议采集监控数据的方法不同，通过 TELNET 协议采集监控数据的方法不支持公钥私钥认证方式，只能通过用户名密码的方式来认证。使用通过 TELNET 协议采集监控数据的方法时，需要注意以下几点：

- 每个通过 TELNET 协议采集监控数据的监控项目，如果被执行的命令序列或脚本程序被分成多行书写，则 Zabbix 系统将其视为多个命令序列来执行，每一行将作为一个独立的命令序列被独立执行，并将所执行的结果作为所采集到数据的一部分，而且每执行一行命令序列所返回的结果也是分行添加到整个监控数据里的。
- 通过 TELNET 协议采集监控数据时，系统支持以 \$、#、>、% 等字符结尾的内容为系统的提示符。换句话说，如果要通过 TELNET 协议采集数据，则 Zabbix 系统用来远程登录被监控主机的用户，在被监控主机上的系统提示符必须要以上述这些字符中的任何一个结尾。否则，Zabbix 系统将无法判断是否已经成功登录到远程主机，从而导致被监控项目因登录错误而不被支持。
- 在执行命令序列所返回的结果中，如果包含以上字符作为结尾的字符串，则 Zabbix 系统会将其从返回结果中除去。但是，当每次采集时所执行的命令序列包含多行时，仅将这样的字符串从第一个被执行命令序列的返回结果中除去，而其他行命令序列的执行结果中这样的字符串不会被除去。这个应该比较好理解，一般当在某个系统中成功执行了某条命令或命令序列时，系统均会自动显示下一行系统提示符。而新显示的系统提示符，作为成功执行命令或命令序列后，系统回显的一部分，当然也会自动被 Zabbix 系统捕获。但是，很显然，系统提示符不应该作为所采集到的数据的一部分，而被视为相应监控项目新采集的监控数据，因此应该将其从返回结果中除去。也正是因为上述原因，Zabbix 系统不鼓励在单个监控项目中执行多行命令序列。
- 如果执行命令序列或脚本程序所返回的值中，包括非 ASCII 和非 UTF8 编码的字符，则在配置监控项目时，应在其关键字里使用编码选项。

2.12 通过 JMX 协议采集监控数据

Java 管理扩展（Java Management Extension, JMX）是一个可作为应用程序、设备和系统等

植入管理功能的框架。简单地说, JMX 框架在不对现有的应用做修改的情况下植入到现有的应用中, 以对现有的应用的运行情况进行管理。或许大家都有一种感觉, 那就是, 一说到框架就有一种云遮雾罩的感觉。其实, 对于管理监控系统来说, 或许不需要管它框架不框架, 只需知道修改一下 Java 程序的启动脚本, 以便在启动应用的同时启动另外一个服务, 而这个服务就是 JMX。我们通过这个 JMX 服务就可以监控 JVM 特性、应用程序运行情况等众多数据。

既然是 Java 管理扩展, 那么毫无疑问, JMX 框架主要是用来采集 Java 容器和 Java 应用程序运行情况的工具。所以, 被监控主机上必须要配置 Java 环境。同时, 通过 JMX 框架采集监控数据是一种被动监控数据采集模式, 即 Zabbix 服务器或其代理服务定时连接到被监控主机的特定端口上, 并采集所需要的数据。具体地讲, 如果能让 Zabbix 系统通过 JMX 框架, 采集被监控主机上 Java 容器或应用程序的数据, 则其流程为: Zabbix 服务器进程 (或 Zabbix 服务器代理进程) 根据用户对监控项目的配置, 周期地将查询请求提交给在本机上运行 (一般都是在本机上运行) 的, 一种被称之为 Zabbix Java 应用程序网关 (Zabbix Java Gateway) 的守护进程, 然后 Zabbix Java 应用程序网关会将这一请求打包成 JMX 格式的请求, 并将其转发到被监控主机的 JMX 服务端口上。接下来, Zabbix Java 应用程序网关会将从被监控主机上接收到的返回信息, 反馈给 Zabbix 服务端进程。Zabbix 服务端进程会将 Zabbix Java 应用程序网关返回的信息, 作为指定主机上相应监控项目的监控数据加以保存。

2.12.1 被监控主机上 JMX 服务的配置

通过上面的介绍, 我们应该已经清楚了, 如果要通过 JMX 协议采集监控数据, 则被监控主机和 Zabbix 服务器 (或服务器代理) 都需要做一些设置。接下来, 我们先来介绍一下如何配置被监控主机上的 JMX 服务。

被监控主机上 JMX 服务的配置相当简单。我们知道, 不管是启动某种 Java 容器也好, 还是启动某个特定的 Java 应用程序, 其实都是执行系统中的 java 命令, 且后面跟一串很长的选项和变量。那么, 要在启动 Java 容器或应用程序的同时启动 JMX 服务, 就只需在原来启动脚本的 java 命令后面再加上一些特定的选项就可以了。例如, 如果不需要安全验证, 则只需将下面这一串选项添加到 Java 容器或应用程序启动脚本的 java 命令之后、原有的启动选项之前就可以了:

```
java \
-Dcom.sun.management.jmxremote \
-Dcom.sun.management.jmxremote.port=12345 \
-Dcom.sun.management.jmxremote.authenticate=false \
-Dcom.sun.management.jmxremote.ssl=false \
```

在启动 Java 容器或应用程序时添加上面这段选项后, 就可以启动 JMX 服务了, JMX 服务的端口号为 12345。或许你会说没有这么简单, 我的应用程序是通过 Tomcat、Jetty、Jboss 等 Java 容器来启动的。而启动这些容器是通过执行类似 startup.sh 的脚本来完成的, 所以我没有办法如此简单地修改 java 命令后面的启动选项。其实配置 Java 容器, 让其启动 JMX 服务的方法也很简单。在这里以 Tomcat 6.0 为例, 介绍一下如何配置 Java 容器的 JMX 服务。

首先, 切换到 Tomcat 容器的 bin 目录下, 即 `cd $TOMCAT_HOME/bin`。然后修改 catalina.sh 脚本文件, 在该文件的 `#!/bin/sh` 行下面添加上面内容并保存:

```
CATALINA_OPTS="-Dcom.sun.management.jmxremote.port=12345 \
-Dcom.sun.management.jmxremote.ssl=false \
-Dcom.sun.management.jmxremote.authenticate=true \
-Dcom.sun.management.jmxremote.password.file=jmxremote.password \
```

```
-Dcom.sun.management.jmxremote.access.file=jmxremote.access \
-Djava.rmi.server.hostname=192.168.5.139 "
```

其中，`-Dcom.sun.management.jmxremote.port` 指定的是服务所侦听的端口号；`-Dcom.sun.management.jmxremote.ssl=false` 行指定不启用 SSL 安全证书；`-Dcom.sun.management.jmxremote.authenticate=true` 行表示启用用户安全认证；`-Dcom.sun.management.jmxremote.password.file=jmxremote.password` 行指定记录用户密码的密码文件为当前路径下的 `jmxremote.password` 文件，也可以用绝对路径指定；`-Dcom.sun.management.jmxremote.access.file=jmxremote.access` 指定记录用户权限的文件为 `jmxremote.access`；`-Djava.rmi.server.hostname` 行指定客户端可以连接的 IP 地址或主机名。需要说明的是，在实际测试中我们发现，虽然用这个方法可以指定哪些 IP 地址或主机名所对应的客户端可以连接到 JMX 服务，但是所指定的端口仍然还是对本机上所有 IP 地址都进行侦听的。另外，启动 JMX 服务时还有一些可用的选项，在这里我们只列出这些选项，但是对于它们的用法和功能不一一赘述，有需要的读者可自行查阅相关资料。

```
-Djavax.net.ssl.keyStorePassword=$YOUR_KEY_STORE_PASSWORD
-Djavax.net.ssl.trustStore=$YOUR_TRUST_STORE
-Djavax.net.ssl.trustStorePassword=$YOUR_TRUST_STORE_PASSWORD
-Dcom.sun.management.jmxremote.ssl.need.client.auth=true
```

好了，接下来需要使用任何一款编辑器在当前目录下创建 `jmxremote.password` 和 `jmxremote.access` 文件，其内容如下。

```
jmxremote.password6587 文件的内容：
monitor password@domain.com
admin adminpassword@domain.com
```

```
mxremote.access 文件的内容：
monitor readonly
admin readwrite
```

接下来，需要将刚刚创建的这两个文件的权限设置成 600，其命令如下：

```
shell> chmod 600 jmxremote.access
shell> chmod 600 jmxremote.password
```

最后，使用 `./startup.sh` 命令启动 Tomcat 容器及其应用。如果在启动过程中没有出现错误，则就可以用 `jconsole` 工具来尝试连接刚刚启动的 JMX 服务了，以确认我们对 JMX 服务所做的配置是正确的，如图 2-12 所示。

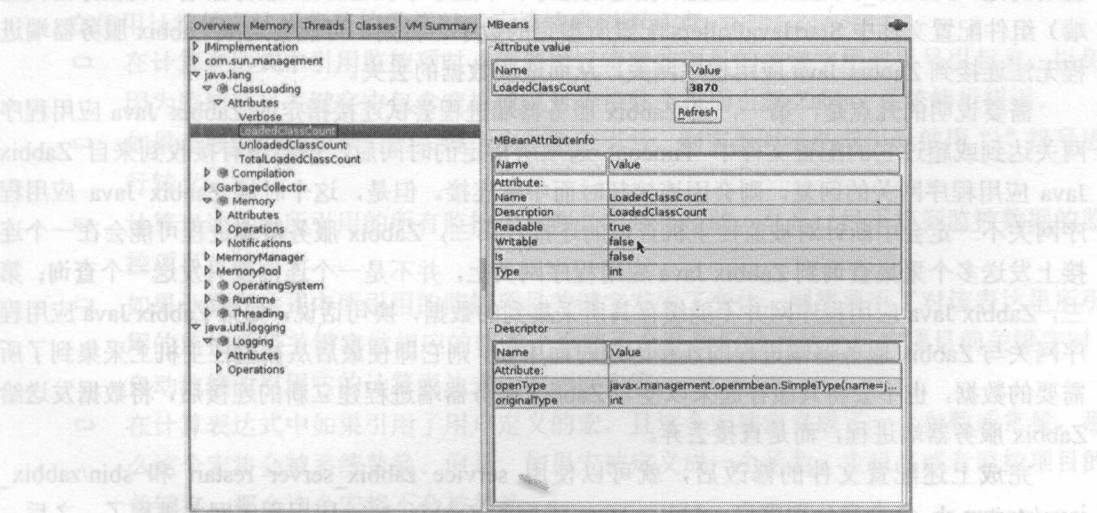


图 2-12 Jconsole 工具截图

2.12.2 Java 应用程序网关的配置

在 Zabbix 服务器上，与 JMX 协议有关的配置也很简单。通过前面对 Zabbix 系统通过 JMX 协议采集监控数据流程的学习，我们知道，Zabbix 系统通过 JMX 协议采集监控数据时，Zabbix 服务器端（或其代理服务器端）进程会将数据采集请求提交到 Zabbix Java 应用程序网关，然后再由 Zabbix Java 应用程序网关向被监控主机提交数据采集请求，最后将采集的结果返回给 Zabbix 服务器端进程。由此可以想到，在 Zabbix 服务器上与 JMX 协议有关的配置主要有两大部分：第一部分是 Zabbix 服务器端进程的配置；另一部分是 Zabbix Java 应用程序网关的配置。

在具体修改配置之前，首先需要确保已经安装了 Zabbix Java 应用程序网关这一组件。如果在编译安装 Zabbix 服务器端组件时，使用了 `--enable-java` 编译选项，则说明已经编译安装了 Zabbix Java 应用网关。如果不记得当初编译安装 Zabbix 服务器端组件时是否使用了 `--enable-java` 选项，那么就切换到 Zabbix 安装目录下的 `sbin` 目录下，看看是否有 `zabbix_java` 目录，和该目录下相应的文件。如果在 Zabbix 安装目录下的 `sbin` 目录下有 `zabbix_java` 目录，且该目录下有多个文件，则说明我们已经编译安装了 Zabbix Java 应用程序网关。

在 Zabbix 服务器端（或服务器代理端）组件的配置文件（`etc/zabbix_server.conf` 或 `etc/zabbix_proxy.conf`）与使用 JMX 协议采集监控数据有关的配置主要有三个选项，它们是：`JavaGateway=IP/hostname`，指定运行 Zabbix Java 应用程序网关进程的服务器 IP 地址或主机名，一般是本机，故该配置项一般配置成 `127.0.0.1`；`JavaGatewayPort=Port`，指定 Zabbix 服务器端进程连接 Zabbix Java 应用程序网关所使用的端口号，默认值为 `10052`；`StartJavaPollers=2`，设定启动多少个进程负载与 Zabbix Java 应用程序网关进行通信。

Zabbix Java 应用程序网关的配置文件为 Zabbix 安装目录下的 `sbin/zabbix_java/settings.sh` 文件。这个配置文件里面的内容很简单：`LISTEN_IP="0.0.0.0"` 指定 Zabbix Java 应用程序网关所侦听的 IP 地址。该配置项如果配置为 `0.0.0.0`，则表示侦听本机上所有网卡上所配置的 IP 地址；`LISTEN_PORT=10052` 配置项指定 Zabbix Java 应用程序网关所侦听的端口号，默认值为 `10052`；`PID_FILE="/tmp/zabbix_java.pid"` 配置项指定启动 Zabbix Java 应用程序网关后所生成的 PID 文件及路径；`START_POLLERS=5` 配置项则设定启动多少个进程具体负责监控数据的采集。需要说明的是，`START_POLLERS` 配置所指定的数字，不应小于在 Zabbix 服务器端（或服务器代理端）组件配置文件中 `StartJavaPollers` 配置所指定的数字。否则，可能会造成 Zabbix 服务器端进程无法连接到 Zabbix Java 应用程序网关，从而造成数据的丢失。

需要说明的几点是：第一，当 Zabbix 服务器端进程尝试连接指定的 Zabbix Java 应用程序网关达到或超过它的配置文件中 `Timeout` 选项所指定的时间后，仍然没有接收到来自 Zabbix Java 应用程序网关的回复，则会因连接超时而中断连接。但是，这个时候 Zabbix Java 应用程序网关不一定会中断针对被监控主机查询的连接；第二，Zabbix 服务器端进程可能会在一个连接上发送多个采集查询到 Zabbix Java 应用程序网关上，并不是一个连接就只发送一个查询；第三，Zabbix Java 应用程序网并不能缓存其所采集到的数据，换句话说，如果 Zabbix Java 应用程序网关与 Zabbix 服务器端进程的连接因超时而中断，则它即使最后从被监控主机上采集到了所需要的数据，也不会将其缓存起来以便与 Zabbix 服务器端进程建立新的连接后，将数据发送给 Zabbix 服务器端进程，而是直接丢弃。

完成上述配置文件的修改后，就可以使用 `service zabbix_server restart` 和 `sbin/zabbix_java/startup.sh` 命令来分别重启 `zabbix_server` 进程和 Zabbix Java 应用程序网关进程了。之后，就可以在 Web 前端页面配置 JMX 类型的监控项目了。关闭 Zabbix Java 应用程序网关进程的命

令是 `sbin/zabbix_java/shutdown.sh`。

2.13 通过计算的方法采集监控数据

通过计算的方法采集监控数据，顾名思义就是将现有的多个被监控项目上所采集的监控数据进行一定的计算，从而获得新的监控数据。因此，这种方法是一种“高级”的监控数据采集方法，是对现有数据的再复合、再利用和再挖掘。从这个角度上来说，通过计算的方法采集监控数据的监控项目都是虚拟的监控项目，不像网卡流量、系统负载等是直接从被监控主机上采集监控数据。虽然，通过计算的方法采集监控数据的监控项目都是虚拟的，但是，在 Zabbix 系统中，它们跟其他实体的监控项目的性质是一样的，即它们的监控数据也是以单独监控项目的方式存在 Zabbix 系统数据库中的，包括历史数据表和趋势数表中的数据；同时，这类虚拟的监控项目也和其他实体类的监控项目一样，可以创建触发器、数据图及使用各种宏变量等。

或许你已经注意到了，在前面章节中已经介绍过通过数据聚合的方法采集监控数据。通过数据聚合的方法采集监控数据与这里所介绍的通过计算的方法采集监控数据在本质上是类似的。它们都是基于现有的监控数据，通过一定的计算方法计算出新的监控数据。但是，这两种监控数据采集方法也有一些差别。例如，数据聚合有额外的数据聚合函数，而通过计算的方法采集监控数据却不支持这类函数。因此，通过数据聚合的方法计算多个主机上的监控数据将会更方便、更灵活；而通过直接计算的方法，对于计算同一台被监控主机上不同监控项目的数据将更实用，也将更简单一些。

既然是通过计算的方法来采集监控数据，那么就少不了计算表达式。通过计算方法采集监控数据的计算表达式语法为：`func(<key>|<hostname:key>,<parameter1>,<parameter2>,...)`。其中，`key` 即为要引用的监控项目的关键字，如果是引用本机监控项目，则可以省略主机名，否则需要写成 `hostname:key` 形式。而 `parameter1`、`parameter2` 等为附加参数。例如，如果要求和某个监控项目最近所采集到的三个监控数据，则 `parameter1` 可以写为 `#3`。附加参数可以省略。同时，在计算表达式中可以支持包括 `last`、`min`、`max`、`avg`、`count` 等函数在内的所有在触发器中可以使用的函数，函数列表见本书的附录 A。

在使用计算的方法采集监控数据时，应该注意以下几点：

- 在计算表达式中引用监控项时，强烈建议将监控项目的关键字用双引号引起来，以免因为监控项目关键字中包含空格或逗号等特殊字符而引起 Zabbix 系统解析错误。
- 如果在被引用监控项目的关键字中包含双引号，则需要对这些双引号使用“\”符号进行转义。
- 计算表达式中所引用的所有监控项目都必须是存在的，且是已经采集到监控数据的监控项目。
- 如果计算表达式中所引用的监控项目关键字发生了变化，则需要手工对该表达式所引用的监控项目关键字做相应的修改。系统不会在我们修改某个监控项目的关键字时，自动地修改引用它的计算表达式里的相应内容。
- 在计算表达式中如果引用了用户定义的宏，且这个宏被定义成了一个参数或常量，那么这个宏将会被系统替换。但是，如果宏被定义成一个函数、主机名或者监控项目的关键字，那么这个宏将不会被替换。
- 如果计算表达式中所引用的监控项目没有找到，或某个函数所引用的参数项没有数据，

或计算表达式中包含被零除的子计算表达式以及计算表达式本身有语法错误, 则该计算表达式所对应的监控项目将不被支持。即虽然相应的监控项目可以被添加进 Zabbix 系统, 但是监控项目的状态为不支持状态。

2.14 本章小结

在本章中, 我们对 Zabbix 系统中可使用的多达 10 多种监控数据采集方法的原理、相关的周边知识及与之相关的 Zabbix 服务器和客户端配置等逐一做了介绍。通过介绍, 可以看出 Zabbix 系统为我们提供的监控数据采集方法是非常丰富的。对于如此众多的监控数据采集方法, 在很短的时间内都掌握其原理并熟练使用它们, 不是一件很容易的事情。有鉴于此, 建议读者从最常用的通过 SNMP 协议采集监控数据和通过 Zabbix 被监控设备代理采集监控数据这两种监控数据采集方法入手, 先熟悉 Zabbix 系统的一般使用。待我们对 Zabbix 系统有了一般性的掌握之后, 再去学习和研究其他的监控数据采集方法。那时, 学习和研究起来可能就要轻松很多。

如前文所述, 本章仅对 Zabbix 系统中所使用到的监控数据采集方法的原理、相关周边知识及与之相关的 Zabbix 服务器和客户端配置做了一些介绍。对于如何通过 Zabbix 系统配置监控项目, 使其能够通过这些监控数据采集方法来采集监控数据并没有做详细的介绍。这部分内容将在后续章节陆续介绍, 并且还需要结合本章所介绍的内容来配置项目。之所以这么编排, 是因为在 Zabbix 系统中涉及的周边知识非常多, 需要先将这些周边知识以及各个监控数据采集方法所使用到的原理讲解清楚, 以免你在具体配置监控项目时产生迷惑。

第3章 Zabbix 系统配置基础

通过前一章的介绍,相信读者对于 Zabbix 系统功能之强大,配置之灵活,应该有一定的印象了吧?然而,当你读完本章后,你也会为 Zabbix 系统提供的非常人性化和方便的管理界面而感叹。当你把 Zabbix 系统搭建和配置完成之后,日常的监控管理和配置都是通过非常漂亮的 Web 界面来完成的。下面就来具体地介绍 Zabbix 系统的 Web 前端页面的使用方法。

3.1 用户登录及创建新用户

很显然,要使用 Zabbix 系统的 Web 前端页面,首先要做的事情就是登录进 Zabbix 系统。下面,首先介绍如何登录 Zabbix 系统,并在此基础上进一步介绍如何在 Zabbix 系统中创建一个新用户。

3.1.1 用户登录

当安装配置好 Zabbix 系统的 Web 前端组件后,安装向导会自动引导我们到系统登录页面上。关于这一点,我们在第1章介绍部署 Zabbix 系统 Web 前端组件时有所介绍。如果用户今后需要访问 Zabbix 系统的 Web 前端组件,只需打开任何一款主流浏览器,并在浏览器的地址栏里输入 Zabbix 系统 Web 前端的访问地址,如 `http://zabbix.domain.com` 或 `http://domain.com/zabbix` 等,就可打开 Zabbix 系统的 Web 前端组件。具体使用哪一种访问地址形式,主要依赖于你有没有针对 Zabbix 系统配置独立的虚拟主机。如果针对 Zabbix 系统 Web 组件配置了独立的虚拟主机,则使用前一种地址形式访问它;否则,用后一种地址形式。当用浏览器打开 Zabbix 系统的 Web 前端组件后,系统首先呈现的是如图 3-1 所示的登录界面。安装完 Zabbix 系统 Web 前端组件后,系统给我们创建的初始超级管理员账号为 Admin,密码为 zabbix。首次登录 Zabbix 系统 Web 前端页面,我们只能使用这个超级管理员账号和密码进行登录。

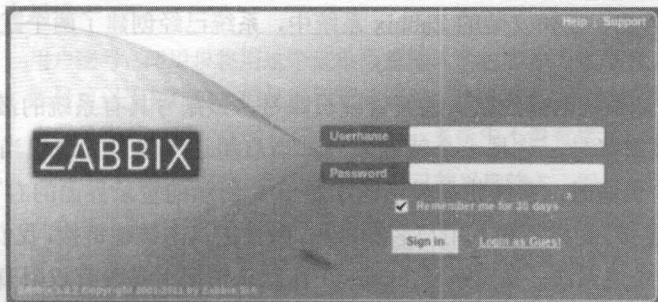


图 3-1 用户登录界面截图

为了防止别有用心的人暴力破解（Brute Force Attacks）Zabbix 系统账号及密码，Zabbix 系统会在用户 5 次尝试登录都失败后，锁定该账号 30s。而且，不管用户账号是否被锁定过，只要某个账号在本次成功登录进系统之前，有过新的登录系统失败的记录，则系统将会显示如图 3-2 所示的最近一次尝试登录失败的信息。

；登陆失败. 最后一次登陆失败发生于 23 一月 2014 10:19

图 3-2 用户登录失败提示信息

3.1.2 创建新用户

刚刚介绍了如何使用系统提供的默认超级管理员账号和密码登录进 Zabbix 系统。但是，超级管理员账号是一个拥有系统最高权限的账号，出于安全方面的考虑，建议在日常管理和学习的过程中，尽可能少用超级管理员账号来操作系统。为此，需要创建一个自己的个性化账号。

依次选择并单击 Web 页面上部的 Administration→User 菜单，系统将打开“配置用户及用户组”页面。在该页面上，系统列出了当前系统中所有用户组的情况。在初安装的 Zabbix 系统中，系统已经创建了如图 3-3 所示的 5 个用户组。

配置用户及用户组

用户组

新建用户组

用户组

正在显示 1 to 5 of 5 已找到

<input type="checkbox"/> 名称	#	成员	状态	WEB前端访问	调试模式
<input type="checkbox"/> Disabled	用户 (0)		禁用	系统默认	禁用
<input type="checkbox"/> Enabled debug mode	用户 (0)		启用	系统默认	禁用
<input type="checkbox"/> Guests	用户 (1)	guest	启用	系统默认	启用
<input type="checkbox"/> No access to the frontend	用户 (0)		禁用	禁用	禁用
<input type="checkbox"/> Zabbix administrators	用户 (2)	Admin, tt	启用	系统默认	禁用

启用选中项

确认 (0)

图 3-3 用户组列表

单击图 3-3 所示的“配置用户及用户组”页面右上角的下拉菜单按钮，选择“用户组”菜单项，系统将列出当前系统中已存在的用户账号信息，如图 3-4 所示。

<input type="checkbox"/> 用户名	名称	姓氏	用户类型	组	是否在线?	登录	WEB 前端访问	调试模式	状态
<input type="checkbox"/> Admin	Zabbix	Administrator	Zabbix 超级管理员	Zabbix administrators	Yes (Wed, 20 Aug 2014 12:46:10 +0800)	Ok	系统默认	禁用	启用
<input type="checkbox"/> guest	Default	User	Zabbix 用户	Guests	No (Sat, 19 Jul 2014 14:47:32 +0800)	Ok	系统默认	启用	启用

图 3-4 用户信息列表

从图 3-4 可以看出，在初安装的 Zabbix 系统中，系统已经创建了两个默认用户：Admin 和 guest。

- Admin。是系统创建的默认超级管理员账号，该账号具有系统的最高权限。
- guest。来宾账号，这是一个特殊账号。当启用这个账号并退出当前用户并再次打开 Web 前台页面时，系统将在显示的如图 3-1 所示的登录界面的右下部显示 Login As Guest 超链接。如果不输入用户名和密码，而直接点击该超链接，我们会以来宾(Guest)的身份登录进系统。来宾账号（Guest）在默认情况下是没有权限查看系统中任何监控信息的，包括监控了哪些主机以及配置了哪些监控项目，等等。但是，Guest 账号也

和其他账号一样，是可以随时被赋予新的权限的。当 Guest 账号被赋予了一定的权限后，它将和普通账号一样，可以查看或修改系统中与其权限相符合的所有信息。但是，使用 Guest 账号登录系统，不需要输入任何账号和密码。而监控系统中的数据往往包含很多敏感信息，例如各类服务器上运行的操作系统类型及版本、Web 服务器的类型及版本等。而这些信息一旦被别有用心的人掌握了，将会对组织内整个信息安全造成很大的威胁。所以，出于安全方面的考虑，如果没有特殊需要，建议平时禁止 Guest 账号登录。

接下来，单击当前页面右上部的新建用户组按钮，系统将打开创建新用户的表单页面，如图 3-5 所示。

The screenshot shows the 'Add user' form in Zabbix. The form is titled 'User' and has tabs for 'Media' and 'Permissions'. The fields are as follows:

- Alias: testuser
- Name: test_username
- Surname: shanghai
- Password: (masked with dots)
- Password (once again): (masked with dots)
- Groups: Zabbix administrators
- Language: English (en_GB)
- Theme: System default
- Auto-login: ☒
- Auto-logout (min 90 seconds): 900
- Refresh (in seconds): 30
- Rows per page: 50
- URL (after login):

At the bottom, there are 'Save' and 'Cancel' buttons.

图 3-5 添加用户表单

图 3-5 所示的添加用户表单中各表单项的含义及其填写要求如表 3-1 所示，读者可参考表 3-1 所列的要求填写该表单。

表 3-1 创建新用户表单项说明列表

表单项	描述
Alias	用户账号，即用户将用这个表单项里输入的内容作为账号登录到系统。虽然登录账号可以支持中文，但是以我们的经验，建议还是尽可能使用英文或汉语拼音作为登录账号。用户账号在同一个Zabbix系统中不得有重复。该表单项为必填项
Name	字面含义为用户名字，你可以根据自己的兴趣输入。但是，因为Zabbix系统是西方人开发的，在西文中，人的姓和名字的书写顺序正好和我们的书写顺序相反，即名在前，姓在后。所以，在这里，可以不必太较真这个表单项名称的字面含义，而根据我们自己的习惯在Name栏中填入姓。这样做，至少在列出用户信息时更符合我们自己的使用习惯。该表单项是必填项

续表

表单项	描述
Surname	字面含义为用户姓，你可以根据自己的兴趣输入。但是，因为Zabbix系统是西方人开发的，在西文中，人的姓和名字的书写顺序正好和我们的书写顺序相反，即名在前，姓在后。所以，在这里，可以不必太较真这个表单项名称的字面含义，而根据我们的习惯在SurName栏中填入名字。这样做，至少在列出用户信息时更符合我们自己的使用习惯。该表单项是必填项
Password	用户密码，必填项
Password(once again)	用户密码确认，为必填项。两次输入的密码必须要一致
Groups	用户所属于的用户组。在Zabbix系统中，用户对被监控资源的访问权限是基于用户组来设置的。所以，用户属于哪些组就决定了它具有访问哪些被监控资源的权限。单击列表框右边的Add按钮，可以为新用户添加新的用户组。一个用户可以属于多个组。当要从列表框中删除某个用户组时，首先在列表框中选择要删除的用户组，一次可以多选，然后单击列表框下边的Delete selected按钮即可。创建用户时，用户至少要属于一个用户组
Language	选择用户在使用Web前端页面时所使用的语言。Zabbix系统支持包括简体中文在内多个国家的语言
Theme	选择用户在使用Web前端页面时所使用的页面风格。当前，Zabbix系统支持“系统默认（System default）”、“经典（Classic）”、“默认蓝色（Original blue）”、“黑色 & 蓝色（Black & Blue）”、“深橘色（Dark orange）”等5种页面风格供用户选择
Auto-login	如果选中，则当用户第一次成功登录后，如果在未来30天内在同一台电脑打开Web前端页面，系统将自动登录进系统，而不需要用户每次都输入账号和密码进行登录。这个功能需要用户的浏览器开启cookie支持
Auto-logout (min 90 seconds)	如果选中该表单项，则用户在登录进系统后，在指定的时间内没有对页面有任何的操作，则系统会自动退出该用户的登录状态。指定时间的最小值不得小于90s
Refresh (in seconds)	指定页面自动刷新的时间间隔。当设置为0时，则表示禁止系统自动刷新页面
Rows per page	设定当页面显示列表类信息时，每张页面所显示的最大信息条数
URL (after login)	指定用户成功登录系统后所跳转到的页面。可不指定。如不指定，则系统默认在用户成功登录系统后跳转到“状态面板（Dashboard）”页面

当按照表 3-1 所列的各表单项的含义填写好适当的信息后，选择当前页面左上部的“消息介质”选项卡，进入用于接收报警信息的“消息介质”页面，如图 3-6 所示。所谓“消息介质”是指系统将通过何种方式给用户发送报警信息。初装的 Zabbix 系统预设了 Email、SMS、Jabber 等 3 种消息介质。关于如何配置新的消息介质，将在后续章节中详细介绍，在这里我们为新增的用户添加 Email 消息介质。

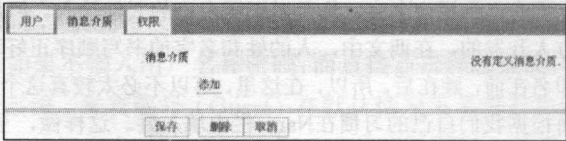


图 3-6 添加用户消息介质

在创建新用户时，系统不会自动为新用户配置默认的消息介质。在配置消息介质页面单击 Add 按钮，系统会弹出配置消息介质对话框，如图 3-7 所示。

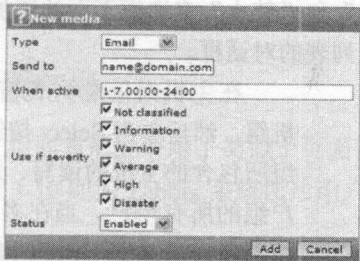


图 3-7 配置消息介质

如图 3-7 所示，从 Type 表单项的下拉菜单里选择 Email 菜单项；在 Send to 表单项里输入用户用于接收报警信息的邮箱地址；When active 表单项里填写该消息介质在哪些时间段被激活使用；Use if severity 表单项则表示对应的消息介质用于发送哪些级别的报警信息，在这里全部选上；而 Status 表单项则用于设定我们所添加的消息介质是启用还是禁用状态，这里当然要选择成启用状态。完成上述信息的配置后，单击 Add 按钮完成消息介质的配置。最后，单击添加用户页面的 Save 按钮，以保存所配置的信息，完成新用户账号的创建。

新建的用户账号在默认情况下是没有任何权限的。接下来，我们要为新建的用户账号赋予一定的权限，该用户账号才能正常使用。

前面提到过，在 Zabbix 系统中，用户的权限是基于用户组来配置的。所以，只需将用户所属于的用户组权限配置正确了，则对应用户组里的所有用户就具有相应的权限了。在前面创建新用户账号时，我们为新创建的用户 testuser 所选择的用户组是 Zabbix Administrators。所以，只需为 Zabbix Administrators 用户组配置所需要的权限，则我们刚刚新建的 testuser 用户账号就自动具有相应的权限了。下面是给用户组配置权限的步骤。

在如图 3-3 所示的用户组列表页面里找到 zabbix administrators 用户组，并单击组名上的超链接，系统将打开用户组配置页面，切换到“权限”选项卡，就可以对 zabbix administrators 用户组的权限进行配置了，如图 3-8 所示。

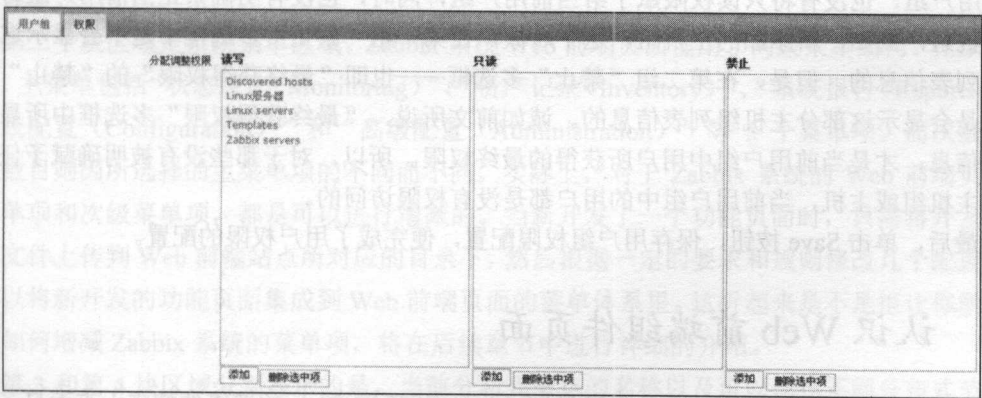


图 3-8 用户组权限配置

在配置用户组权限的选项卡里，系统给我们呈现出了两大组各三列的多选框。基于篇幅的限制，我们只截取其中的一组多选框，其他多选框的样式与图 3-8 显示的样式是类似的。这三

列多选框分别显示的是选定的用户组对其具有读写、只读和禁止访问权限的主机组和主机（包括模板，在 Zabbix 系统中，模板也被视作主机看待）及主机组信息列表。

分别单击“读写”、“只读”和“禁止”多选框下面的添加按钮，均会弹出如图 3-9 所示的能显示当前系统中所有主机组列表的对话框。

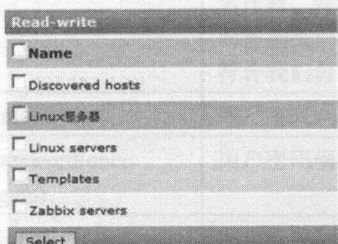


图 3-9 主机组信息列表

在主机组信息列表窗口里单选或多选需要添加的主机组，然后单击 Select 按钮，系统将会把所有选中的主机组所包含的主机的读写、只读或禁止权限赋予属于当前用户组的所有用户。当在多选框中选中的一个或多个主机组，然后单击多选框下面的“删除选中项”按钮，则可以把所选中的主机组所包含的主机的相应权限从当前用户组中删除。

在配置用户组权限选项卡上的第一组多选框下面，还有一组多选框——它是用来显示用户组最终所获得权限（Calculated permissions）的。最终获得权限多选框有两行，分别用来显示当前用户组已经被赋予读写、只读和禁止访问权限的主机组或主机列表。这两行多选框里所显示的主机组或主机列表信息，是系统根据当前用户组被赋予的权限计算出来的、该用户组可能所拥有的最终权限的主机组列表和主机列表。可以使用这两行多选框显示出来的信息，确认和核对对当前用户组赋予的权限是否正确。现在，我们看到的最终获得权限多选框是两行，这是因为这里的 Zabbix 系统被配置成独立服务器模式。实际上，如果将 Zabbix 系统部署为分布式模式，则最终获得权限多选框是三行，另外一行用于显示当前用户组所具有的分布式节点权限列表信息。

细心的读者可能已经发现了一个很有趣的现象，在这两大组多选框中都有一列是“禁止”列。可是奇怪的是，有时候在第一组“禁止”多选框里所显示的列表信息与第二组（也即最终获得权限组）“禁止”多选框中所显示的列表信息并不完全一样，而且第二组所显示的列表信息要比第一组所显示的列表信息多。这是什么原因呢？难道是 Zabbix 系统中的一个 Bug 吗？原来，在第一组“禁止”多选框里显示的只是明确被禁止的主机组的列表。但是，还有一些主机组或主机从来没有明确地给当前用户组赋予任何权限。即没有将这些主机组的读写权限赋予给当前用户组，也没有将只读权限赋予给当前用户组，同时，也没有明确禁止当前用户组访问这些主机组，例如，新添加的主机组。这个时候，在第一组“禁止”多选框中是不会显示这些主机组列表信息的。但是，在第二组“禁止”多选框——也即“最终获得权限”的“禁止”多选框中是会显示这部分主机组列表信息的。诚如前文所说，“最终获得权限”多选框中所显示的权限信息，才是当前用户组中用户所获得的最终权限。所以，对于那些没有被明确赋予任何权限的主机组或主机，当前用户组中的用户都是没有权限访问的。

最后，单击 Save 按钮，保存用户组权限配置，便完成了用户权限的配置。

3.2 认识 Web 前端组件页面

通过前面的介绍，实际上我们已经见识过 Zabbix 系统 Web 前端组件的庐山真面目了。但是，我们对于这个 Zabbix 系统所提供的 Web 前端组件的各个功能模块的作用还不是很熟悉。下面就来具体地介绍一下，我们能通过这个 Web 前端组件做些什么事情。在下面的叙述中，我们有时候也称 Web 前端组件为图形用户界面（Graphical User Interface, GUI）或直接简写它

的英文缩写 GUI，还有可能在不至于产生混淆的情况下，直接将 Web 前端组件简称为 Web 页面。读者在阅读本书时，只需将这几种不同的叫法视做同一个东西即可，而不必为它到底叫什么而困惑。

3.2.1 Web 前端组件页面布局

Zabbix 系统图形用户界面在页面布局上使用了类似于页面框架的结构。所以在 GUI 中几乎所有页面的布局都是一样的。图 3-10 即为 Zabbix 系统图形用户界面的页面布局情况。

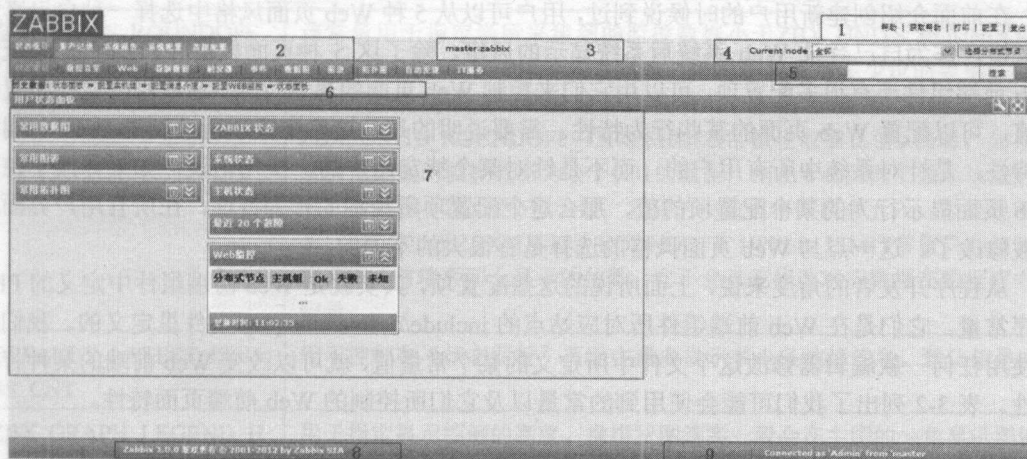


图 3-10 Zabbix 图形用户界面页面布局图

Web 组件的页面布局如图 3-10 所示，第 1 区为快速连接区，主要显示了 Zabbix 系统官方文档、当前登录用户配置及退出等超链接信息。对应这些超链接，读者只需亲自单击一下，打开对应的页面，就可以看到它们主要是做什么用途的了，在这里不一一赘述。但是，需要说明的是，当当前登录用户开启了“调试”功能时，则在这个区域将会多出一个“调试”超链接，它主要是用户用于调试 Web 前端组件的。具体如何使用 Web 前端组件的调试功能，将在后续章节详述。

第二个块区域是系统菜单区域。Zabbix 系统 Web 前端页面使用了两级菜单结构。在默认情况下，主菜单包括“状态统计 (Monitoring)”、“资产记录 (Inventory)”、“系统报告 (Reports)”、“系统配置 (Configuration)”和“高级配置 (Administration)”等 5 个菜单项。而次级菜单项的数目则因所选择的主菜单项的不同而不同。实际上，对于 Zabbix 系统的 Web 前端页面的主菜单项和次级菜单项，都是可以进行增减的。当新开发了一个功能页面时，只需将开发好的页面文件上传到 Web 前端站点所对应的目录下，然后根据一定的要求和规则修改几个配置文件，就可以将新开发的功能页面集成到 Web 前端页面的菜单体系里。这听起来是不是很让你激动？具体如何增减 Zabbix 系统的菜单项，将在后续章节中进行详细的介绍。

第 3 和第 4 块区域分别显示的是，当前分布式节点的名称以及可以选择不同分布式节点的下拉菜单。如果 Web 前端组件被配置成独立服务器模式，则这两个区域将不会显示任何内容。

第 5 块区域显示的是全局搜索表单。通过它，可以很方便地搜索当前系统中已经配置的主机信息。

第 6 块区域很像我们经常在许多网站上看到的网站路径导航条。但是，它显示的不是网站路径导航信息，而是我们之前访问过的页面记录。它在某些时候可以给我们的工作带来方便，大大提高我们的工作效率。

第 7 块区域为数据区。这个区域的内容会依据我们所选择的菜单项的不同而不同。

第 8 和第 9 块区域分别显示的是系统的版权信息和当前登录用户的信息。

3.2.2 Web 前端组件行为配置

在前面介绍创建新用户的时候说到过，用户可以从 5 种 Web 页面风格中选择一种自己喜欢的风格，作为自己登录 Web 系统后系统显示的风格。除了这 5 种页面风格外，Zabbix 系统的 Web 前端组件还有很多配置项，可以用它们来控制 Web 页面的显示行为。通过修改这些配置项的值，可以配置 Web 页面的某些行为特性。需要说明的是，这些配置项所控制的 Web 前端行为特性，是针对系统中所有用户的，而不是针对某个特定用户的。换句话说，如果修改了控制 Web 页面显示行为的某个配置项的值，那么这个配置项所控制的行为特性，在所有用户界面上都被修改了。这一点与 Web 页面风格的选择是有很大的不同的。

从程序开发者的角度来说，上面所说的这些配置项，其实就是 Web 前端组件中定义的 PHP 程序常量。它们是在 Web 前端组件所对应站点的 include/defines.inc.php 文件里定义的。我们只要使用任何一款编辑器修改这个文件中所定义的某个常量值，就可以改变 Web 前端的某种行为特性。表 3-2 列出了我们可能会使用到的常量以及它们所控制的 Web 前端页面特性。

表 3-2 控制Web前端行为特性的常量列表

常量名称	描述
ZBX_PERIOD_DEFAULT	指定显示数据图时默认显示的时间跨度。通俗地说，就是当通过Web页面打开某个被监控主机上的某个监控项目的数据图时，系统默认为我们显示多长时间的数据图，单位是秒。这个常量的默认值为3600，这也是所能设置的最小值。如果这个常量的值被设置成比3600还小，那么Web页面将以一小时的时间跨度来显示数据图。而且，这个常量的值不能被设置成比接下来将要介绍的ZBX_MIN_PERIOD常量值还要小。否则，在Web页面上查看数据图时将会收到警告信息，且ZBX_MIN_PERIOD常量所设置的值将不起作用
ZBX_MIN_PERIOD	定义数据图所能显示的最小时间跨度。单位是秒，默认值为3600，即1小时。这个值也是该常量所能设置的最小值。当这个常量所设置的值小于3600时，系统将以一小时的时间跨度来显示数据图，该常量将不起作用。当用户在数据图滚动条上所选择的数据图显示时间跨度小于这个常量所定义的时长时，系统将会报错，且数据图将无法显示
ZBX_MAX_PERIOD	定义数据图能显示的最大时间跨度，单位是秒，默认值为63 072 000，即两年
GRAPH_YAXIS_SIDE_DEFAULT	指定在显示数据图时，Y坐标轴是显示在数据图的左边还是右边。当取值为0时，则Y坐标轴显示在数据图的左边；当取值为1时，则Y坐标轴显示在数据图的右边。默认值为0
ZBX_UNITS_ROUNDOff_THRESHOLD	指定用于控制显示数据位数的阈值。这个常量将和接下来介绍的两个常量ZBX_UNITS_ROUNDOff_UPPER_LIMIT 和ZBX_UNITS_ROUNDOff_LOWER_LIMIT配合一起使用。详细说明请参阅接下来将要介绍的两个常量的含义说明。这个常量的默认值为0.01

续表

常量名称	描述
ZBX_UNITS_ROUNDOFF_UPPER_LIMIT	该常量用于指定当所采集到的监控数据大于ZBX_UNITS_ROUNDOFF_THRESHOLD常量所指定的数据时，采用小数点多少位的数据进行显示。该常量的默认值是2。例如，当该常量取默认值2，且ZBX_UNITS_ROUNDOFF_THRESHOLD常量也取默认值0.01时，如果系统所采集到的监控数据为大于0.01的数据，则在Web页面上显示该数据时，只显示整数部分和小数点后两位小数，之后的小数位将做四舍五入
ZBX_UNITS_ROUNDOFF_THRESHOLD	该常量用于指定当所采集到的监控数据小于ZBX_UNITS_ROUNDOFF_THRESHOLD常量所指定的数据时，采用小数点后多少位的数据进行显示，该常量的默认值是6。例如，当该常量取默认值6，且ZBX_UNITS_ROUNDOFF_THRESHOLD常量也取默认值0.01时，如果系统所采集到的监控数据为小于0.01的数据，则在Web页面上显示这些数据时，会显示小数点后6位小数，之后的小数位将做四舍五入。需要说明的是，该常量以及ZBX_UNITS_ROUNDOFF_UPPER_LIMIT常量只会控制数据在Web前端页面上显示的位数，它不会真正修改对应数据在数据库中的记录
DEFAULT_LATEST_ISSUES_CNT	用于控制在“状态面板”页面中最多显示多少条故障信息。默认值是20
ZBX_GRAPH_LEGEND_HEIGHT	用于指定显示图例的高度。像饼状图等图一般会在主图的一角显示图例。如果这个常量值被指定得过小，而实际图例的高度要远远大于这个常量所指定的值的话，则图例可能会显示不完整。这个常量的默认值是120

3.2.3 Web 前端组件维护模式配置

Zabbix 系统中的 Web 前端组件可以被设置成维护模式，以临时禁止用户对 Web 前端页面的访问。这个功能在有较多用户访问和使用 Zabbix 系统的 Web 前端页面时比较有用。当 Zabbix 系统管理员打算对 Zabbix 系统的数据库进行维护时，他可以提前将 Zabbix 系统的 Web 前端组件设置成维护模式。这样，就可以禁止其他用户在数据库维护期间，对系统中所配置的监控信息进行修改，从而引发数据不完整。当 Zabbix 系统的 Web 前端组件处于维护模式下时，它可以允许某些指定 IP 地址的用户访问它，而禁止其他 IP 地址的用户对它的访问。要配置 Zabbix 系统 Web 前端组件为维护模式，只需修改 Web 前端组件站点所对应目录下的 conf/maintenance.inc.php 文件即可。对 maintenance.inc.php 文件的修改也比较简单，只需将下面这三行的注释符，即//符号去掉，并做相应的修改即可。

```
1. //define('ZBX_DENY_GUI_ACCESS',0);
2. //$ZBX_GUI_ACCESS_IP_RANGE = array('127.0.0.1');
3. //$REQUEST['warning_msg'] = 'Zabbix is under maintenance.';
```

上面这三条语句的含义是：当将常量 ZBX_DENY_GUI_ACCESS 的值修改为任何一个非零的数字（即将上述第一行中括号中的 0 修改成任何一个非 0 的数字）时，即表示开启 Web 前端组件维护模式。第二行，\$ZBX_GUI_ACCESS_IP_RANGE 常量，定义了为维护期间可以允许访问的客户端 IP 地址列表。例如，当将括号中的 IP 地址 127.0.0.1 修改成 192.168.5.139 这个 IP 地址时，则表示在 Web 组件维护期间，允许来自 IP 地址为 192.168.5.139 的用户的访问。而当

将这个常量所定义的 IP 地址修改成 192.168.5.1-254 时,则表示在 Web 组件维护期间,允许客户端 IP 地址属于 192.168.5.1 到 192.168.5.254 范围内的任何一个 IP 地址的用户的访问。第三行则定义了,当用户试图在 Web 组件维护期间访问 Web 页面时,系统给用户的提示信息。例如,当将这行修改成 `$_REQUEST['warning_msg'] = '当前我们监控系统正在维护,暂停所有用户的访问'` 时,如果用户在 Web 组件维护期间访问 Web 页面,则系统将会显示出如图 3-11 所示的提示信息。

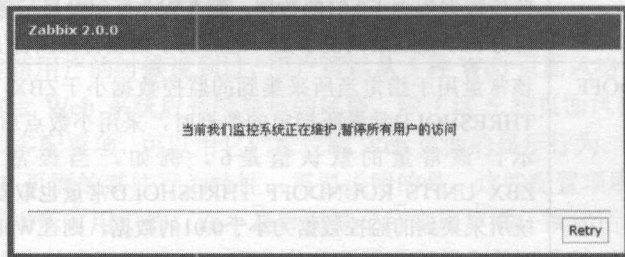


图 3-11 Web 前端维护期间提示信息

3.3 Zabbix 系统菜单项主要功能

在前面介绍图形用户界面页面布局时提到过,Zabbix 系统图形用户界面菜单系统在默认情况下是由“状态统计(Monitoring)”、“资产记录(Inventory)”、“系统报告(Reports)”、“系统配置(Configuration)”和“高级配置(Administration)”等 5 个主菜单项和若干个次级菜单项组成的。接下来简要地说明一下这 5 个主菜单项以及它们各自子菜单项都能用来做什么操作。

3.3.1 “状态统计”菜单项的功能

“状态统计”,英文为 Monitoring,这个菜单项主要是用来显示系统中所采集到的各种监控数据、触发器、事件以及数据图等信息的。该主菜单项下包含“状态面板(Dashboard)”、“数据总览(Overview)”、“Web”、“最新数据(Latest data)”、“触发器(Triggers)”、“事件(Events)”、“数据图(Graphs)”、“图表(Screens)”、“拓扑图(Maps)”、“自动发现(Discovery)”和“IT 服务(IT Services)”等 11 个子菜单项。

提示: 因为不同版本的 Zabbix 系统汉化效果不完全一样,因此,你在系统中所看到的菜单项中文名称和在这里所介绍的菜单项中文名称可能不完全相同。但将要介绍的各菜单项所对应的英文名称应该与你从系统中所看到的对应菜单项的英文名称是一致的。之所以出现这种情况,主要原因是因为汉化造成的。因此,在接下来的介绍中,对于页面上所输出的中文信息,我们将尽可能将对应的英文信息也写出来。所以,当发现本书叙述中所给出的中文信息与你在实际系统中所看到的中文信息不一致时,应以本书中给出的英文信息为准,可与你从实际系统中所看到的英文信息进行比对。

1. “状态面板(Dashboard)”菜单项的功能

正如这个子菜单项的中文名称所揭示的那样,状态面板页面显示的是,系统中比较重要的

信息摘要。在这个页面上，显示了当前 Zabbix 服务器的运行状态（如 Zabbix 服务器端进程是否正常运行）、最近的 N 个故障信息（对于 N ，我们在前面章节中介绍过。它由控制 Web 前端行为的 PHP 常量——DEFAULT_LATEST_ISSUES_CNT 所控制，默认值是 20。可以通过修改这个常量的值，来控制状态面板页面显示的最近故障数量）、按主机组分组显示的主机状态统计信息（如正常设备多少台，故障设备多少台）、各种收藏夹（如常用数据图收藏夹、常用图表收藏夹和常用拓扑图收藏夹等）、系统状态信息（如按主机组分组的灾难问题多少、严重问题多少，等等）和 Web 监控以及常用的数据图、图表收藏夹，等等，如图 3-12 所示。

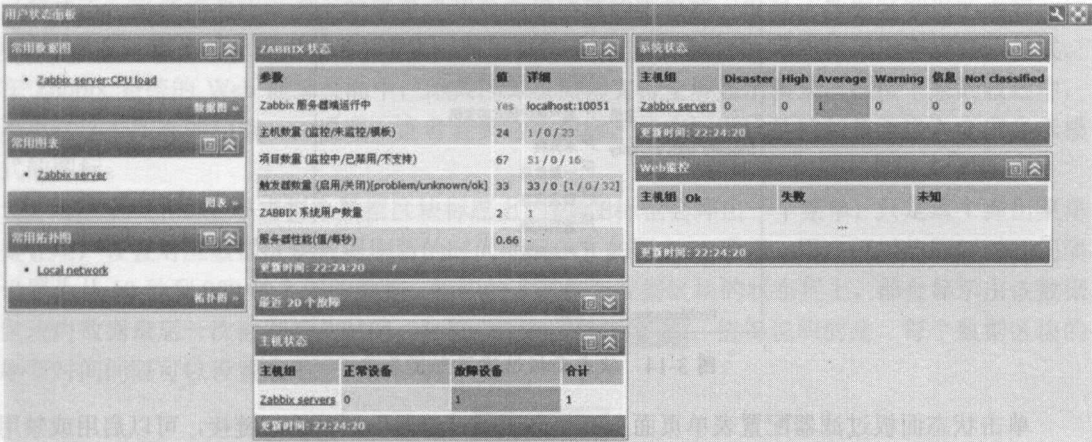




图 3-12 状态面板截图

从图 3-12 可以看出，整个状态面板页面的数据域被分成三列，各列由显示不同统计信息的区块组成。实际上，对于这些区块，可以用鼠标单击区块的标题部分，然后按住不放，就可以将其进行拖动，并可以将其放到所希望的位置上。因此，对于状态面板页面上的数据区块布局用户可以根据自己的喜好进行调整。单击各数据区块标题栏的  图标，可以自由地收缩或展开该区块所显示的内容。

如图 3-12 所示，可以将经常使用的“数据图”、“图表”和“拓扑图”等图表信息添加到各自的收藏夹里，这样可以方便我们今后使用。操作方法是：用鼠标单击各收藏夹标题栏上的  图标，系统将会弹出如图 3-13 所示的菜单。选择这个弹出菜单中的“添加”或“移除”菜单项，系统就会弹出一个新窗口页面，以供用户选择所需要添加的数据图项目；或弹出下一级菜单，以供用户选择要移去的项目。

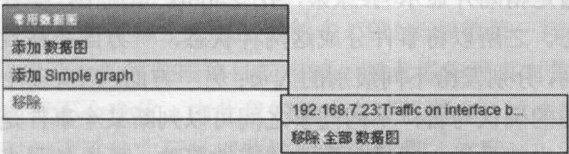



图 3-13 收藏夹弹出菜单

对于在状态面板上显示的“系统状态”信息、“主机状态”信息以及最近故障信息等，可以设置过滤器，以便将一些我们并不关心的或者不需要重点关心的信息给过滤掉，使状态面板上只显示我们非常关心的关键信息，从而让该页面所显示的内容更有重点。设置状态面板过滤器的操作方法是：用鼠标单击“用户状态面板”工具条右边的  图标，系统会弹出过滤器配置

页面，如图 3-14 所示。

状态面板过滤器 启用

主机组 ☒ 已选中

组

- Discovered hosts
- Linux servers
- Zabbix servers

添加 删除选中项

主机 ☒ 显示维护模式的主机

Triggers with severity

- ☒ 未分类
- ☒ 一般信息
- ☒ 警告信息
- ☒ 重要问题
- ☒ 严重问题
- ☒ 灾难问题

Problem display 全部



图 3-14 状态面板过滤器配置表单


单击状态面板过滤器配置表单页面上“状态面板过滤器”后面的超链接，可以启用或禁用状态面板过滤器。当状态面板过滤器被禁用后，系统中所有主机、触发器和故障信息都会被统计显示在状态面板页面上的对应区域内。在这个表单里，“主机组”表单项很容量理解。这个表单项有两个单选项：“全部”和“已选中”，“全部”当然是指所有主机组的主机；而当选“已选中”选项后，系统将显示“组”多选框，单击这个多选框下面的“添加”或“删除选中项”按钮，就可以对该多选框中显示的主机组列表执行增减操作。


接下来的“主机”和“Triggers with severity”表单项也很简单，只需勾选选中项目前面的多选框就可以了，而且各项的含义也很简单，只需看字面意思就能知道这几个表单项的含义。故对于这两个表单项，在这里就不作过多的叙述。

重点要说明的是 Problem display 表单项，这个表单项给我们提供了“全部”、“分隔的”和“仅未确认的”这三个选项。对于“全部”这个表单项的含义我们很好理解，即所有的事件都被包含在统计对象内；但是，对于“分隔的”和“仅未确认的”这两个选项，仅从字面意思上看，我们理解起来可能就有问题了，至少笔者在第一次配置这两个选项时就比较迷惑。什么叫“分隔的”？难道是指隔开显示吗？原来，在 Zabbix 系统中，事件有两种状态，分别为已确认状态或未确认状态。之所以将事件分成这两种状态，一方面是在设置报警条件时，可以根据这两种状态进行区别，分别发给不同级别的人员；另一方面，当有多个同事维护同一套 Zabbix 系统时，通过事件状态的确认与否，不同同事之间可以判断某个事件是否已经有其他同事在处理或者已经处理完成。也就是说，通过对事件状态的修改，可以加强多个同事之间相互协作的能力。这样的话，就比较容易理解了，所谓“分隔的”在这里实际上是指，将已确认的和未确认的事件分开统计，然后分别显示在状态面板页面上。例如，当 Problem display 表单项选择为“全部”选项时，则事件统计数据在状态面板页面的系统状态区域就会显示成如“6”这种形式；而当将这个表单项选择为“分隔的”选项后，同一个对象的数据格式就显示为“2 of 6”的形式。所以，这里这个“分隔的”选项名可能是因为由英文翻译成中文不准确造成的误解。笔者认为，

这个选项的名称翻译成“分开显示”可能更符合实际一些。通过上述的分析，读者对于最后一个选项“仅未确认的”的含义应该不难理解了吧？它其实就是表示，只统计并显示没有被确认的事件的数量。

当配置并保存状态面板过滤器的设置后，刷新状态面板页面，就可以立即看到“用户状态面板”工具条右边的图标由变成。所以，今后只需看“用户状态面板”工具条上的这个图标，就可以直接判断出当前状态面板是否已经启用过滤器。

而当单击用户状态面板工具条最右边的图标时，系统将进入到全屏显示模式。在全屏显示模式下，系统将关闭页面上的菜单栏和页面最底部的版权栏，仅显示数据区域里的内容，这样，可以在一屏里显示更多的监控数据。而当再次单击上述图标时，系统将退出全屏显示模式。在 Zabbix 系统的 Web 前端页面中，绝大部分页面都支持全屏显示模式。在接下来的叙述中，不再对这个图标的功能一一说明，读者看到这个图标，就应知道这是控制是否进入全屏显示模式的图标。

与收藏夹相似，单击每个数据区块标题上的图标也会弹出一个菜单，只是这个弹出菜单是让用户设置对应数据区块数据刷新的时间间隔的，如图 3-15 所示。用户可以将刷新时间间隔设置为从 10 秒到 900 秒之间的数据。相应的，在每个数据区块的状态栏上，都会显示出该数据区块内数据最后一次被刷新的时间，如更新时间: 22:37:15。值得说明的是，每个数据区块的刷新时间间隔可以设置成不一样的，并不要求统一。

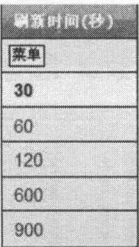


图 3-15 状态面板弹出菜单

对于状态面板页面上绝大多数数据区块内显示的数据，只需看它对应的对象名称，就能很快理解和判定对应数据区块内显示的是哪类对象数据。所以，在这里就不对每个数据区块内的各个数据项做一一说明。这里，我们重点要介绍一下 Zabbix 状态栏内的“服务器性能（值/每秒）”这一项数据的含义。或许你跟我一样，首次看到这个项目名称的时候，觉得这个项目的数值反应的是 Zabbix 服务器在之前的某段时间内，平均需要处理的监控数据的条数。也就是说，这个项目反映的是服务器历史负载情况。如果是这样的话，那么，这个值应该是会有变化的，而且变化的幅度可能还比较大，类似于 Linux 系统中 CPU 负载数值一样的变化。但是，实际情况却是，如果 Zabbix 系统没有任何监控项目的增减，或者说你没有对你的 Zabbix 系统做配置上的修改，那么，这个值就不会变化。由此我们基本上可以断定，这个数据并不是反映 Zabbix 系统当前或者历史一段时间内系统实际处理监控数据的数量。实际上，这个项目的值是 Zabbix 系统根据你系统中的配置，比如说被监控的主机数量、监控项目的数量等，然后根据一定的算法推算出来的，当前 Zabbix 系统可能需要每 ns 处理的监控数据的数量。也就是说，它反映的是一个期望值，而不是历史实际值。那或许你会问，既然它不是历史实际数据的反映，那我要这个数据有什么用呢？虽然它不是历史实际数据的反映，但是它的大小反映出了当前 Zabbix 系统对服务器硬件要求的高低。一般来说，在我们 Zabbix 系统刚投入使用时，被监控主机和监控

项目的数量都相对较少，这个时候 Zabbix 系统对服务器硬件的要求也比较低。而随着被监控主机的增加，监控项目的数量就可能会快速增长。通过关注这个数据的变化，可以提醒我们是不是该给 Zabbix 服务器提升硬件性能了。另外需要说明的一点是，这个项目的数据虽然反映的是预计 Zabbix 服务器平均每秒要处理的新的数据的条数。但是，Zabbix 服务器每处理一条新采集的数据，对数据库的查询操作却是多次的。所以，即使这个数据有较小的变化，也有可能给服务器的负载，特别是磁盘 I/O 的读写，带来很大的需求和压力。

读者可能还记得，我们在介绍 Zabbix 系统 Web 前端组件的安装时，提到过在安装 Zabbix 系统 Web 前端组件时，可能需要修改 php.ini 文件。PHP 软件安装包里为我们所提供的默认的 PHP 配置文件，满足不了安装 Zabbix 系统 Web 组件的要求。如果不做相应的修改，在安装 Web 组件时，安装向导就不让我们执行下一步操作。现在，如果因为某种原因，重新编译安装了 PHP 软件，或者修改了 php.ini 文件，以使 PHP 配置没有达到 Zabbix 系统 Web 组件的最低安装要求，则当打开状态面板时，在 Zabbix 状态栏的最下面就会显示如下所示的相应错误提示：

PHP memory limit 64M Minimum PHP memory limit is 128M (configuration parameter "memory_limit")

2. “数据总览(Overview)”菜单项的功能

“状态统计”的第二个子菜单项是“数据总览(Overview)”。顾名思义，数据总览页面显示的是，当前系统中所有被监控主机和每个监控项目最近一次所采集到的监控数据以及触发器状态的情况总览，如图 3-16 所示。

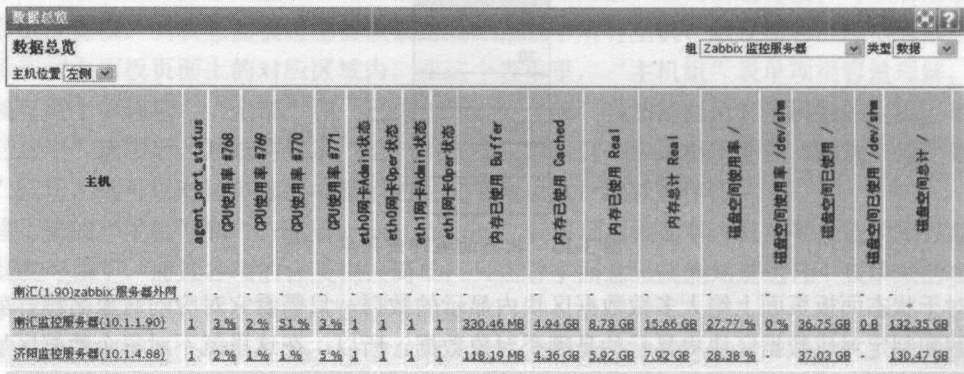


图 3-16 “数据总览”截图

从图 3-16 可以看出，在数据总览页面的左上部有一个“主机位置”下拉菜单。这个菜单总共有两个菜单项，分别为“顶部”和“左侧”。当选择“顶部”菜单项时，页面将会以横排的方式显示主机名，而被监控项目的名称则纵排。而当选择“左侧”菜单项时，显示方式则正好相反，即横排不同的监控项目名称，而纵排主机名。图 3-16 就是选择“左侧”菜单项后页面的显示效果。不管你是选择“顶部”还是“左侧”菜单项，页面顶部横向排列的主机名或监控项目的名称都是竖排的，且在 2.0.4 以前的 Zabbix 版本中，这部分内容都是被系统生成图片显示在页面上的。相对应的，图 3-16 中最左列显示的主机名或监控项目名称，则不管你选择的是哪种显示方式，它们都是以文本的方式显示在页面上。

我们再往右看，会看到一个叫做“组”的下拉菜单。对这个下拉菜单不需要做过多的说明，大家应该知道这是让我们选择哪个主机组的数据将被显示在页面上。但是，需要说明的是，如果系统中所监控的主机或监控项目比较多的话，我们应尽量少选中这个下拉菜单中的“全部”

菜单项。因为，如果选择“全部”菜单项，将会导致系统生成的数据量非常大，从而使我们打开页面的速度非常慢，甚至使浏览器经常性地失去响应。

数据总览页面上显示的内容可以是“数据”和“触发器状态”两种类型之一。图 3-16 显示的就是“数据”类型的数据。此时，页面上所显示的内容是，被选定主机组所对应所有主机上所有监控项目最新采集到的监控数据。在有数据的单元格里，当单击鼠标左键时，系统会弹出如图 3-17 所示的弹出式菜单。

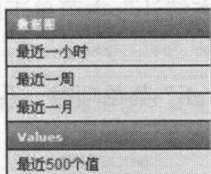



图 3-17 数据总览页面弹出菜单

由图 3-17 所示的弹出式菜单,我们可以快速地查看对应监控项目最近一个小时、最近一周、最近一个月和最近 500 个值的监控数据。

而当选择显示“触发器状态”类型数据时，系统也会为我们呈现出与“数据”类型类似的总览信息列表。只是，在原来显示数据的单元格里，现在显示的是代表触发器状态的、一个个有着不同颜色的小方块，如图 3-18 所示。这些小方块的不同颜色，代表了对应触发器当前所处的状态。当小方块在不停地闪烁时，则表示对应触发器的状态最近一次改变不超过 30 分钟（默认值是 30 分钟，这个时间长度可以自己设定，后续章节将介绍如何设置这个时间长度）。

需要更新的组件列表						数据总量
项目	5秒	10秒	30秒	1分	5分	10分钟以上
Zabbix agent	0	0	0	0	0	0
Zabbix agent(主动方式)	0	0	0	0	0	0
Simple check	0	0	0	0	0	0
SNMPv1 agent	0	0	0	0	0	0
SNMPv2 agent	0	0	0	0	0	0
SNMPv3 agent	0	0	0	0	0	0
Zabbix external	0	0	0	0	0	0

图 3-18 数据总览(触发器状态)截图

而如果想知道上述的这些小块的不同颜色各代表什么含义,则可以单击数据总览标题栏上的图标,此时系统会弹出一个图例窗口,在该窗口中显示了不同的颜色代表了何种触发器状态,如图 3-19 所示。

与选择“数据”类型时一样，当在触发器状态总览页面的有颜色方块上单击鼠标左键时，系统也会为我们弹出一个弹出式菜单，如图 3-20 所示。

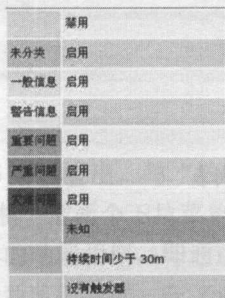


图 3-19 触发器状态图例截图



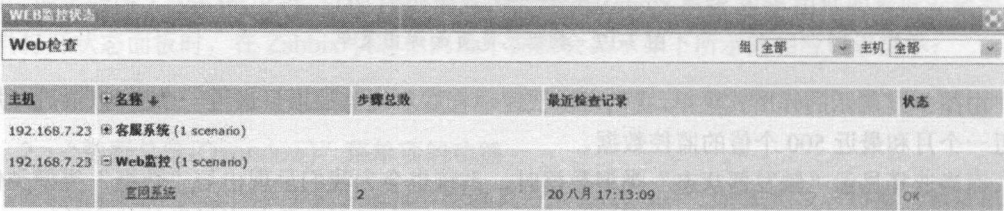
图 3-20 数据总览弹出式菜单(触发器状态)

由图 3-20 所示的弹出式菜单，我们可以快速地查看与相应触发器相关的事件或对该触发器的状态进行确认，以及快速查看与该触发器相关的监控项目的数据图等。

3. “Web” 菜单项的功能

通过 Zabbix 系统，可以对指定的 Web 站点进行监控，并能对指定 Web 站点不同页面的下载速度和下载耗时进行记录。如何配置 Zabbix 系统对指定站点进行监控，我们将在后续章节做详细介绍。这里首先介绍一下，“状态统计”主菜单下的“Web”菜单项都能给我们提供什么样的功能。

当单击“状态统计”菜单下的“Web”菜单项时，系统将显示出如图 3-21 所示的 Web 监控状态统计信息总览页面。

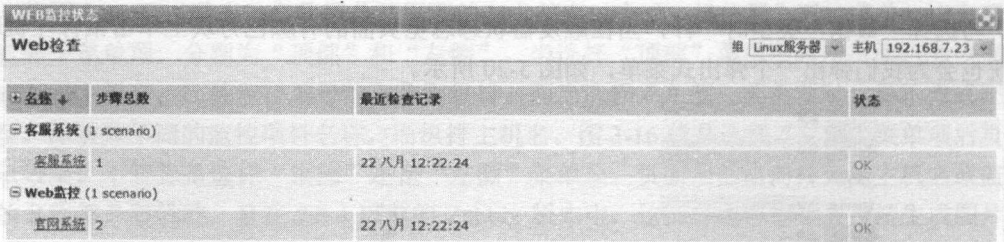


主机	名称	步骤总数	最近检查记录	状态
192.168.7.23	客服系统 (1 scenario)			
192.168.7.23	Web监控 (1 scenario)			
	客服系统	2	20 八月 17:13:09	OK

图 3-21 Web 监控状态总览

在图 3-21 所示的 Web 监控状态总览页面上，系统除了提供“组”和“主机”两个下拉菜单，以供我们选择要查看的主机组和指定主机的 Web 监控状态信息外，还为我们显示了“主机”，即对应 Web 监控是依附在哪台主机上的（实际上，Web 站点的监控并不一定就要配置在该站点真实部署的那台主机上）；“名称”，则是指 Web 站点监控所在的分类名称；“步骤总数”则是指对应场景总共有多少个检查“步骤”；“最近检查记录”，当然就是指最近一次检查的时间信息了；而最后一列的“状态”则表示，对应场景最近一次检查的状态是否正常。

当单击分类名称前面的 ⊕ 号时，系统将展开属于这个分类的所有 Web 监控场景的列表。例如如图 3-21 中的“客服系统”就是场景名。单击场景名上的超链接，系统将会把我们导向到显示该场景详细信息的页面上。在详细信息页面上，包含一个包括该场景所有步骤最近一次下载速率、响应时间、响应代码和状态等详细信息列表（如图 3-22 所示），以及该场景下所有步骤的响应时间和下载速率数据图，分别如图 3-23 和图 3-24 所示。



名称	步骤总数	最近检查记录	状态
客服系统 (1 scenario)			
客服系统	1	22 八月 12:22:24	OK
Web监控 (1 scenario)			
客服系统	2	22 八月 12:22:24	OK

图 3-22 Web 监控场景详细信息列表

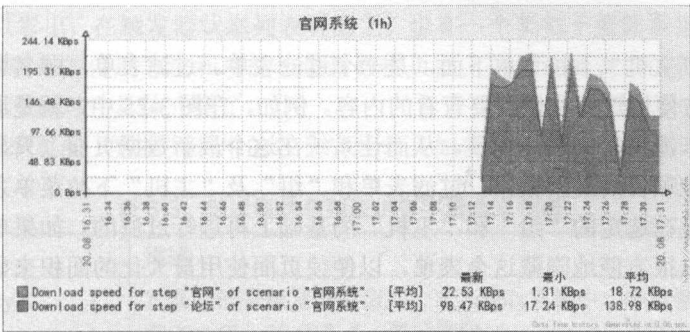


图 3-23 Web 监控场景下载速度数据图

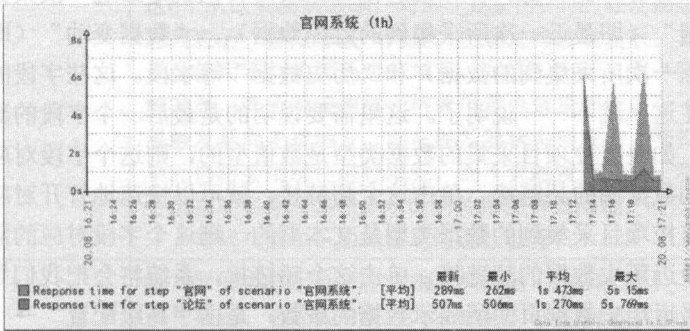


图 3-24 Web 监控场景响应时间数据图

4. “最新数据(Latest data)”菜单项的功能

“状态统计”主菜单下的第 4 个子菜单项是“最新数据(Latest data)”子菜单项。通过该子菜单项，可以查询指定主机组和指定主机的监控项目所采集到的最新数据、最近一次监控数据的采集时间、上一次所采集到的监控数据与最近一次所采集到监控数据之间的变化情况等信息，如图 3-25 所示。

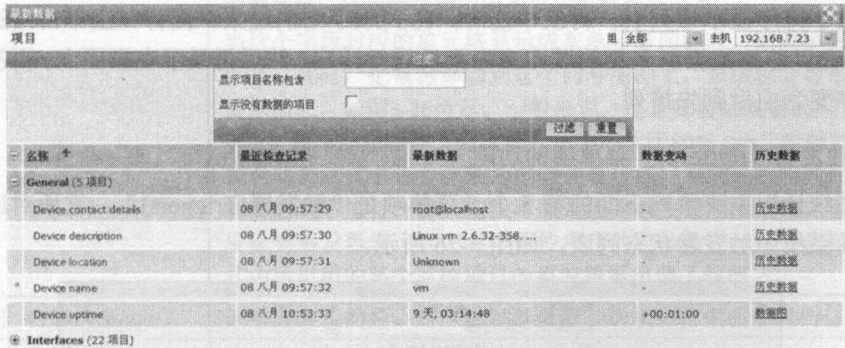






图 3-25 最新数据页面截图

从图 3-25 可以看出，与“数据总览”和“Web”页面类似，在最新数据页面的数据区也有“组”和“主机”这两个下拉菜单。它们的作用当然也和“数据总览”等页面上的“组”和“主机”下拉菜单的功能相似，即通过这两个下拉菜单，可以很方便地选择我们要查看的主机组和具体主机的最新监控数据。在 Zabbix 系统的 Web 前端的许多页面上都有类似的这样一组下拉菜单，其功能也都是相同的。所以，在后面章节的叙述中，对于这组下拉菜单的功能不再一一

叙述。

在“组”和“主机”下拉菜单下面，是一个过滤表单。过滤表单，顾名思义，就是通过这个表单，可以很方便地过滤我们需要查看的内容。例如，在图 3-25 中，就是通过过滤监控项目名称中包含“网卡流量”的监控项目，从而让系统在这个最新数据页面上只显示我们最关心的网卡流量这类监控项目的最新数据。过滤表单和“组”及“主机”下拉菜单之间是与的关系，也即过滤的对象是在选定的“组”和“主机”的基础上再进行过滤的。如果单击这个过滤表单的标题栏，则可以很方便地隐藏这个表单，以便使页面使用最大化的面积来显示我们所关心的数据。

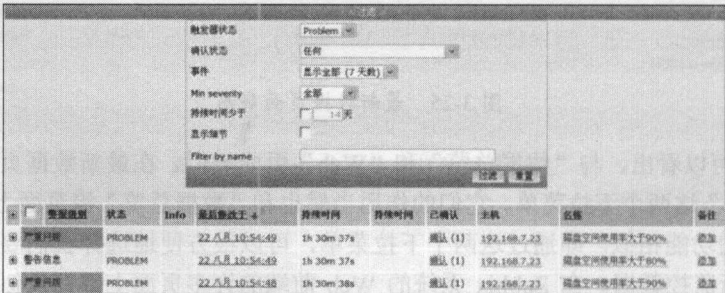
最近新数据列表页面上的数据包括“主机”（即为主机名称）、“名称”（笔者认为这个字段应该翻译成“分类”比较好）、“最近检查记录”（也就是最近一次采集监控数据的时间点）、“最新数据”（即最近一次所采集到的监控数据）、“数据变动”（即为最近一次所采集到的数据减去上一次所采集到的数据）和“历史数据”等字段。这些字段的含义应该还是比较容易理解的，在这里就不一一说明了。这里需要说明的是最后一个字段的数据，即“历史数据”字段的数据。如果监控项目采集的数据类型是数值型的，则这个字段对应的监控项目上就会显示一个连接到数据图的超链接。单击这个超链接，就可以快速地打开对应监控项目的简单数据图。而如果监控项目采集到的数据类型是文本型的，则这个字段对应的监控项目上，将会显示一个连接到最近历史数据的超链接。单击这个超链接，系统将会把我们导向到最近历史数据页面。然而，最近历史数据页面所显示数据的条数，是由“高级配置”→“常规”页面上的“搜索/过滤 元素限制”配置项控制的。

需要说明的是，从图 3-25 中可以看出，在这个页面的主机列前也都有  图标。很显然，这个  图标与之前在介绍 Web 子菜单项时所介绍的  图标的功能是一样的。即单击它，系统就会展开所隐藏的监控项目；而再次单击它，系统将再次隐藏它们。另外，当单击标题栏上的  图标时，系统会展开所有的被隐藏的监控项目。

当单击最新数据列表标题栏上的不同字段时，系统将会按所单击的字段进行排序显示列表内容。例如，当在标题栏上单击“主机”字段时，则系统在显示该列表内容时，将会按照主机名字符的大小排序显示该列表；再次单击时，排列顺序则倒过来。同样，当单击最近检查记录时，系统将会以监控数据所采集的顺序由先到后进行排序，然后再显示在页面上；再次单击时，则排列顺序变为由后到先排列。

5. “触发器(Triggers)”菜单项的功能

“状态统计”主菜单项下面的第 5 个子菜单项是“触发器(Triggers)”，它所对应的页面用于显示当前系统中触发器状态列表，如图 3-26 所示。



触发器状态									
触发器状态		Problems							
确认状态		任何							
事件		显示全部 (7 天数)							
Min severity		全部							
持续时间少于		12 天							
显示细节									
Filter by name									
		过滤 重置							
类型	状态	Info	最后修改	持续时间	持续时间	已确认	主机	触发	备注
严重问题	PROBLEM	22.8.10:54:49	1h 30m 37s	确认 (1)	192.168.7.23	磁盘空间使用率大于90%	激活		
警告信息	PROBLEM	22.8.10:54:49	1h 30m 37s	确认 (1)	192.168.7.23	磁盘空间使用率大于80%	激活		
严重问题	PROBLEM	22.8.10:54:48	1h 30m 38s	确认 (1)	192.168.7.23	磁盘空间使用率大于90%	激活		

图 3-26 触发器状态列表页面截图

从图 3-26 可以看出，在触发器状态列表页面上，也有一个类似于最新数据页面上的过滤器表单。这个过滤器表单的使用方法和功能与最新数据页面上的过滤器表单的使用方法和功能是一样的，在这里就不再对它具体介绍了。

触发器状态列表页面上数据列表中的各个字段含义如表 3-3 所示。

表 3-3 触发器状态列表中各字段含义列表

字段名称	描述
警报等级 (Severity)	它显示的是触发器的级别。在 Zabbix 系统中，触发器可以分为“未分类(Not Classified)”、“一般问题(Information)”、“警告问题(Warning)”、“重要问题(Average)”、“严重问题(High)”和“灾难问题(Disaster)”这6个从低到高的级别。对应于这6个不同的触发器级别，在触发器状态列表的“警报等级”列里，当触发器被触发后，其对应单元格的背景会使用不同的颜色进行填充，如图3-26所示的“警告信息”级别的触发器，其所对应的单元格就用黄色背景颜色进行了填充。但是，如果触发器尚未被触发，则在触发器状态列表的“警报等级”列的对应单元格将会用绿色进行填充，而不管触发器是属于哪个级别。触发器各级别的名称及其在显示时所使用的背景颜色，可以通过“高级管理”→“常规”菜单下的相应功能进行设置
状态 (Status)	触发器的状态分为正常(OK)和问题(PROBLEM)两种状态。对于这两种状态，笔者认为称之为未触发（也就是OK状态）和已触发（也就是PROBLEM状态）可能更合适一些，至少不会像使用“正常”和“问题”这两种状态称谓一样，会引起初学者的误解。所以，本书在后面的叙述中经常称触发器的状态为已触发或未被触发。与“数据总览”页面中所显示的触发器状态类似，在触发器状态页面上，如果触发器的状态最近一次改变到显示时的时长不超过30分钟（默认值是30分钟，这个时长可以通过“高级管理”→“常规”页面来设置），则状态字段对应的单元格里显示的内容将会闪烁。另外，如果触发器从问题状态转变为正常状态的时间不超过30分钟，则即使过滤器里被设置成只显示问题状态的触发器，这类触发器的信息也还会被显示出来
信息(Info)	当这个字段对应的单元格显示成灰色问号图标时，则表示系统有提示信息。此时，把鼠标移动到这个问号图标上，系统就会显示出相应的提示信息，如图3-26所示。一般来说，这意味着触发器的配置有问题。例如，图3-26所示的“磁盘空间使用率大于80%/sys”的触发器，系统提示其表达式中包含除数为零的子表达式，系统无法将其代入到触发器表达式中进行计算，这很可能是因为触发器刚刚创建，系统还没有采集到触发器表达式中某个监控项目的监控数据，或者是触发器表达式中所引用的某个监控项目的监控数据出现了问题
最后修改于 (Last Change)	该字段显示的是，对应触发器最近一次状态发生改变时的日期和时间
持续时间 (Age)	该字段显示的是，对应触发器最近一次状态发生改变到页面显示时的持续时间，也就是对应触发器在当前状态下已经持续的时长
已确认 (Acknowledged)	该字段显示的是，对应触发器的状态是否已经被确认。如果对应触发器的状态已经被确认，则对应触发器的该字段内容会用绿色字体的“已经确认”显示；如果对应触发器的状态从未确认过，则系统会用红色字体显示一个“确认”超链接，单击这个超链接，我们将会被系统导航到触发器状态确认页面

续表

字段名称	描述
主机 (Host)	该字段显示的是触发器所对应的主机名称。在主机名称上单击鼠标，系统会弹出如图3-27所示的弹出式菜单。通过这个弹出式菜单，可以针对相应主机执行系统中定义好的脚本程序。如果要在系统中定义脚本程序，可以依次选择“高级配置”→“脚本”菜单项，在“配置脚本”页面上进行相应配置。同时，通过这个弹出式菜单，可以快速查看对应主机的最新数据和主机图表等信息
名称 (Name)	本字段显示的是触发器的名称。在触发器名称上单击鼠标，系统同样会弹出一个弹出式菜单，如图3-28所示。通过这个弹出式菜单，可以快速连接到对应触发器的配置页面，打开与该触发器相关的事件显示页面，显示与该触发器相关联的监控项目数据图及打开在配置该触发器时指定的Web页面等
备注 (Comments)	如果在配置触发器时，在“描述”字段里填写了内容。那么系统就会在这个字段中显示一个“显示”超链接。通过这个超链接，可以查看这个触发器被设置的备注是什么内容，并可以修改。如果对应触发器在配置时没有填写“描述”字段内容，则系统在这个字段里会显示“添加”超链接，单击这个超链接，可以给相应触发器添加备注信息



图 3-27 主机名称上的弹出式菜单截图

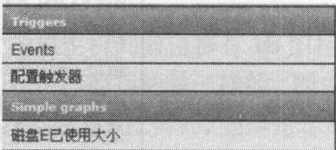


图 3-28 触发器名称上的弹出式菜单截图

关于触发器状态列表页面上的过滤器，在默认情况下，仅显示处于“问题”状态以及状态最近从“问题”状态转变为“正常”状态下的触发器信息。可以通过过滤器表单中的“触发器状态”下拉菜单，来修改显示触发器的范围。但是，修改只能对本次显示有效，当下次再次打开“触发器状态”页面时，系统默认仍然只显示处于“问题”状态和最近从“问题”状态转变为“正常”状态触发器的列表信息。

6. “事件(Events)”菜单项的功能

“状态统计”下的“事件”子菜单项所对应页面如图 3-29 所示。

与其他“状态统计”下子菜单项所对应的页面类似。在“最新事件”页面上系统也提供了“组”和“主机”下拉菜单，以供我们选择和过滤想查看事件的内容。但是，除此之外，系统还另外提供了事件来源 (Source) 下拉菜单，以供我们按照事件的来源来选择所需要查看的事件信息。同时，在历史事件标题栏的右边，系统还提供了一个“导出到 CSV 文件”按钮。单击这个按钮，可以将当前页面上所显示的事件信息，导出到一个 CSV 格式的文件中。

细心的读者可能已经发现，“最新事件”页面上的过滤器与前面所介绍的“最新数据”以及“触发器状态”页面上的过滤器不太相同。“最新事件”页面上的过滤器中多了一个如图 3-30 所示的用于指定过滤时间范围的时间滚动条。

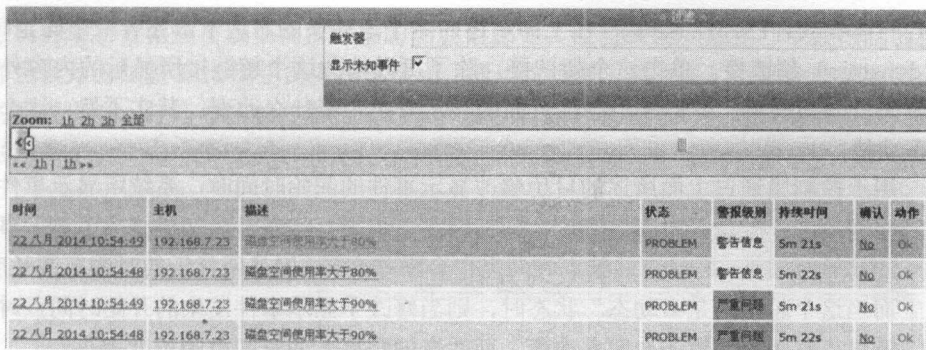


图 3-29 最新事件(触发器)截图

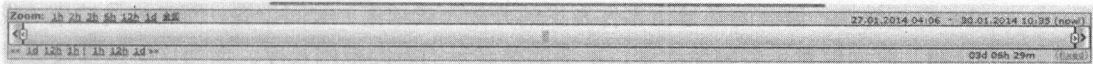


图 3-30 时间滚动条

通过图 3-30 所示的时间滚动条，可以比较方便地指定显示指定时间范围内的事件信息。大概你已经注意到了，在这个滚动条的左边的上部有如 Zoom:1h、2h、1d、12h 等超链接，这些超链接是用来选择，在多长时间范围内产生的事件将显示在页面上。例如，当单击“1h”超链接时，如果结束时间选择成当前，那么系统就将显示最近一个小时内所产生的事件信息；相应的，如果单击“1d”超链接，则表示显示最近一天内所发生的事件信息。而在这个滚动条的左下部，也显示有“<< 1m 1w 1d 12h 1h | 1h 12h 1d 1w 1m >>”等字样的超链接。在这些超链接中，当分别单击“|”分隔符前面的 1m、1w、1d、12h、1h 等超链接时，表示修改显示的时间范围向前推一个月、一个星期、一天、12 小时和 1 小时等。例如，如果当前显示的是今天 12:05~13:05 所产生的事件，那么当单击一次“1h”超链接时，则系统将显示从 11:05 开始所产生的事件信息。而“|”分隔符后面的 1h、12h、1d 等超链接，与前面部分超链接的功能是类似的，只是它修改的是时间范围的结束时间点。例如，如果当前显示的是昨天 12:05~13:05 之间所发生的事件信息，那么当单击“1h”超链接时，则时间范围的结束时间点就从 13:05 变成 14:05。从这个小小滚动条的功能上，是不是可以看出 Zabbix 系统非常人性化？然而，不仅如此，实际上当拖动时间滚动条，或者拉长或缩短这个时间滚动条时，也可以选择我们想要查看事件所发生的时间范围的起始点。而在这个滚动条的右上部，则显示了当前所显示数据的起止时间点。当分别单击这个起止时间点上的超链接时，系统会弹出如图 3-31 所示的日历窗口。通过这个日历窗口，可以很方便地定义所要显示的数据的起止日期和时间。



图 3-31 日历窗口截图

细心的读者或许已经注意到了,在上面所述的这个滚动条的右边下部还有一个固定(fixed)/动态(dynamic)超链接。单击这个超链接,除了可以修改这个超链接所显示的内容外,似乎并不能带来页面的刷新,即似乎不会给页面的显示效果带来什么改变。其实不然,这个超链接具有以下功能:

- 用于控制当通过上面所述的日历修改显示事件的起始时间时,系统所显示事件的发生时间范围长度是否改变。当这个超链接是“固定”状态时,如果修改了显示事件发生的开始时间点,则对应的结束时间点也会跟着改变,但是所显示的时间范围长度不变。而当这个超链接是“动态”状态时,则当修改了显示事件发生的开始时间点时,对应的结束时间点位置不会跟着改变,而改变的是显示的时间范围长度大小。
- 当上面所述的这个超链接处于“固定”状态时,如果单击滚动条左边下部的 1h、12h、1w 等超链接,修改的是显示事件发生的开始(或结束)时间位置,而对应的结束(或开始)时间位置也会随着改变,不变的是显示的时间范围长度。而当这个超链接处于“动态”状态时,如果做相应的操作,则效果刚好相反,即显示的时间范围长度将会改变,而起止时间位置不变。
- 在这个滚动条上还有◀和▶两个按钮,通过这两个按钮,可以改变页面所显示事件的起始时间位置和结束时间位置。而上面所述的这个超链接的状态,对这两个按钮的行为也有着如上面两点所述类似的控制。即,当这个超链接的状态为“固定”状态时,单击这两个按钮,同时改变显示事件的起始时间位置和结束时间位置,不变的是显示的时间范围长度;而当这个超链接处于“动态”状态时,则显示的时间范围长度改变,对应的结束时间位置和起始时间位置不变。

在后面将要介绍的“数据图”等页面上,系统也将提供类似的时间滚动条。其功能和“最新事件”页面上的时间滚动条的功能是类似的,到时候就不再详述了。

“最新事件”数据列表中所显示的字段数和字段名称,会因为我们所选择事件来源类型不同而不同。例如,当选择事件的来源类型为触发器时,则数据列表中显示的字段包括:“时间”,表示事件发生的时间;“主机”,表示该事件来源于哪台主机,这里显示的是主机的名称;“描述”,则是指产生该事件的触发器的名称;“状态”,则表示事件所对应触发器当前的状态,即“问题”状态还是“正常”状态。与前面介绍触发器状态的时候,所介绍的触发器的状态类似,如果事件所对应的触发器状态转变成当前状态的时间不超过 30 分钟,则状态字段里的内容将会以闪烁的方式显示;“报警级别”,则是指事件所对应触发器的级别;“持续时间”,则是指该事件从产生到现在所持续的时长;“确认”,显示超链接,单击这个超链接,系统将会把我们导航到触发器状态确认页面;“动作”,表示该事件是否有与之相匹配的报警或执行远程命令的动作。如果有,则这个字段内将显示对应动作的执行状态。动作的执行状态包括 OK,即所有与该事件相匹配的动作都已成功执行完毕。“正在进行”,表示系统正在执行与事件相匹配的动作。而如果这个字段内显示的是“-”,则表示对应事件没有匹配到任何动作。

当单击触发器型事件的时间字段上的超链接时,系统将打开对应事件的详细页面,如图 3-32 所示。

在图 3-32 中,事件的详细页面显示了事件的来源(如事件所来源的主机、触发器等)、触发器级别、触发器表达式、事件发生的详细时间以及该事件动作的执行情况等内容。

当用鼠标单击某个事件“主机”字段上的超链接时,系统同样也会弹出如图 3-27 所示的弹出式菜单。通过这个弹出式菜单,可以针对特定的主机执行系统中预先定义脚本程序,或查

看该主机的最新数据及与该主机所对应的图表信息等。而当单击“描述”字段内超链接时，系统同样也会弹出如图 3-28 所示的弹出式菜单。通过这个弹出式菜单，可以快速地连接到对应触发器的配置页面，打开与该触发器相关的事件显示页面，查看与对应触发器相关联的监控项目的数据图等。

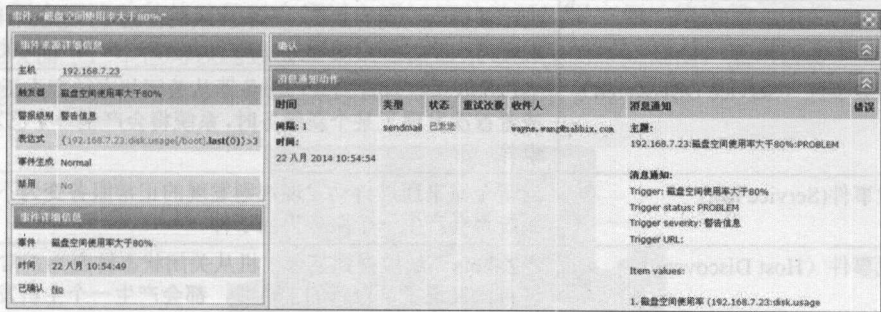


图 3-32 事件详细信息

而当查看“自动发现”类型最新事件时，系统将会显示如图 3-33 所示的最新事件列表信息。由图 3-33 可以看出，自动发现类型的事件与触发器类型的事件，在显示的字段和内容上略有不同。与触发器类型的事件相比，自动发现类型的事件在显示字段上有如下差异：“IP 地址”，显示的是 Zabbix 系统通过自动发现而找到的新的被监控主机的 IP 地址；“主机域名”，则显示的是系统通过反向 DNS 查询，而获取到的新发现的主机所对应的主机名；“描述”则显示的是发现了某台主机还是某个服务，如果发现的是某个服务，则这里显示的是发现主机所使用的服务的名称信息；而“状态”列则显示的是被发现的主机或某个具体服务的当前状态。

时间	IP地址	主机域名	描述	状态
30 一月 2014 11:34:19	192.168.10.3		主机	DOWN
30 一月 2014 11:34:19	192.168.10.4		主机	LOST
30 一月 2014 11:34:21	192.168.10.7		服务: ICMP ping	DOWN
30 一月 2014 11:34:21	192.168.10.9		服务: ICMP ping	LOST
30 一月 2014 11:34:13	192.168.10.25		主机	UP
30 一月 2014 11:34:15	192.168.10.49		服务: ICMP ping	UP
30 一月 2014 11:33:42	192.168.10.50		主机	UP
30 一月 2014 11:33:45	192.168.10.69		服务: ICMP ping	UP

图 3-33 最新事件列表（自动发现）

前面介绍过，当系统自动发现一个主机或服务时，都会产生一个事件。具体来说，就是系统在配置并开启了一个或多个自动发现规则时，自动发现功能在如表 3-4 所示的情形下都会产生相应的事件信息。

表 3-4 产生自动发现型事件的情形列表

事件	描述
服务开启事件（Service up）	每当Zabbix系统检测到某台主机上有新的服务开启时，都会产生一个服务发现事件
服务关闭事件（Service down）	当某台已被自动发现的主机上，之前检测已开启的某个服务，在本次检测中发现其已处于不可用状态时，系统就会产生一个服务关闭事件

续表

事件	描述
主机开启事件(Host up)	当系统检测到某台已被自动发现的主机上,至少开启了一个服务时,系统会产生一个主机开启事件
主机关闭事件 (Host Down)	当已被自动发现的主机,在本次系统检测时,系统未在其上发现任何一个已开启的服务,则系统会产生一个主机关闭事件
已发现服务事件 (Service Discovered)	当Zabbix系统检查到某台服务器从关闭状态转变为正常状态,或者首次发现了某个新服务时,系统将会产生一个已发现服务事件
服务失效事件(Service lost)	当某个原来通过自动发现规则发现的正常服务变为不正常时,系统将会产生一个服务失效事件
主机发现事件 (Host Discovered)	当Zabbix系统检查到某台主机从关闭状态转变为正常状态,或者首次发现了某台新的主机时,都会产生一个主机发现事件
主机失效事件 (Host lost)	当Zabbix系统通过自动发现功能发现了某台正常的主机后,该主机的状态随后又转变为非正常状态,系统会产生一个主机失效事件

7. “数据图(Graphs)”菜单项的功能

在 Zabbix 系统中,当某个监控项目在配置时,其数据类型被配置成数值型(包括无符号整型和浮点型),则系统会自动为这个监控项目创建对应的简单数据图(需要由“数据总览”或“最新数据”页面进入简单数据图页面查看)。但是,Zabbix 系统自动创建的简单数据图是基于单个监控项目的。比如说,我们监控某台主机网卡的流量,一般我们对这块网卡的进出流量数据分别采集。也就是说,在配置监控项目时,对于这块网卡的发送流量和接收流量,实际上是分别配置监控项目的。所以,此时如果要查看这块网卡的流量图,且仅使用系统提供的简单数据图功能,则需要分别查看这块网卡的接收流量图和发送流量图,没有办法在一张数据图上同时查看这块网卡的接收流量图和发送流量图。而同时,Zabbix 系统也提供了创建自定义数据图的功能。如果需要配置某个被监控主机的数据图,则可以在配置这台被监控主机的监控项目或者在创建模板时进行配置。具体如何配置自定义数据图,将在后续章节中详细介绍。

通过“状态统计”→“数据图”子菜单项下的页面,可以查看系统中所定义的自定义数据图。当选择该菜单项后,系统将显示所选定主机上定义的数据图,如图 3-34 所示。

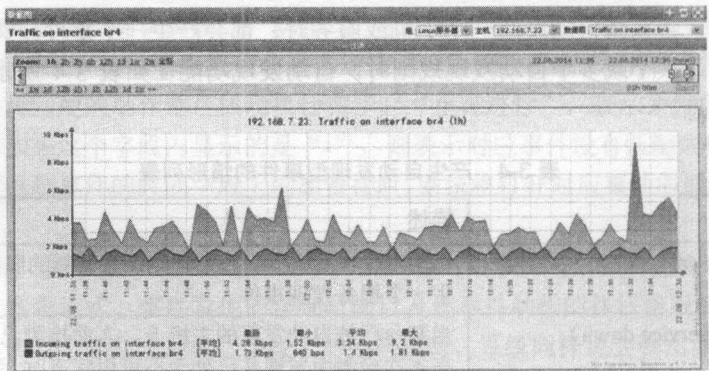


图 3-34 数据图

在数据图页面上同样包含“组”、“主机”、“数据图”等下拉菜单以及带有滚动条的过滤器，这些组件的功能与我们在前面章节中介绍其他页面时，所介绍的对应组件的功能是类似的。因此，这里就不再一一介绍了。读者如果还有不清楚的地方，请参阅本书前面相关章节的内容。

在前面介绍滚动条的功能时介绍过，通过滚动条可以很方便地修改显示数据的时间范围。在数据图页面中，不但可以通过滚动条来修改显示数据图的时间范围，还可以在数据图上按住鼠标左键拖动，选择所需显示的数据图的时间范围，如图 3-35 所示。

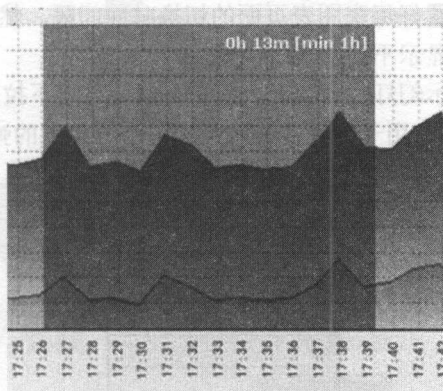





图 3-35 选取数据图显示的时间范围截图

在数据图页面中，还有几个之前没有看到过的图标，它们的功能和作用分别是：

图标 ：将当前数据图添加到“状态统计”→“状态面板”页面上的“收藏夹”中。通过这个收藏夹，可以快速地打开需要经常访问的数据图。

图标 ：重置当前数据图的显示设置为默认设置。即，显示最近一个小时的数据图。

图标 ：开启或关闭数据图的全屏显示模式。在全屏显示模式下，系统将隐藏掉菜单栏、导航栏和状态栏中的内容，仅显示数据图信息。

8. “图表 (Screens)” 菜单项的功能

图表 (screens)，从这个名字可以猜到，在图表页面上既可以显示“图”，即数据图等，也可以显示表格。在 Zabbix 系统的图表页面中，几乎可以汇总前面介绍的所有内容。例如，系统自动生成的简单数据图、我们自己定义的数据图、事件信息汇总、Zabbix 系统状态信息汇总、各主机组的触发器等信息，都可以将它们放到一张图表页面中，如图 3-36 所示。

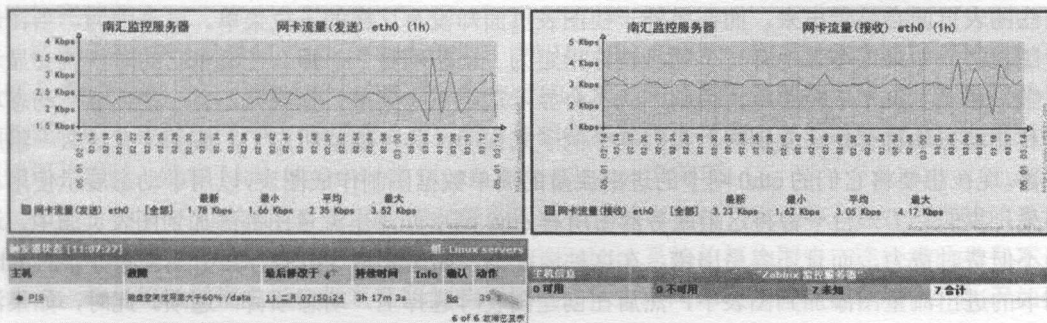



图 3-36 图表页面截图

在图表页面标题栏的右部有一个下拉菜单，这个下拉菜单用于选择页面所显示的内容是图表还是幻灯片。那么幻灯片又是什么东西呢？相信使用过微软 PowerPoint 软件的朋友应该对幻灯片的概念不陌生。其实，Zabbix 系统中幻灯片的概念和微软 PowerPoint 软件中幻灯片的概念是类似的。在某些场合，需要将数据图、系统状态信息等内容动态展示出来，比如说，在很多 IDC 机房里，都有那种挂在墙上的用于动态展示系统实时监控信息的监控屏。这个时候，如果这个展示屏上所显示的内容完全靠手工来切换的话，不但费时费力而且几乎没人能做到 24 小时不停地切换。在这种情况下，就需要使用幻灯片的功能了。可以把多个图表页面配置在一起，形成一组幻灯片页面，并设置好每张图表页面的切换时间间隔。然后，浏览这组幻灯片时，系统就可以做到自动切换，并显示不同图表页面的内容了。

当在图表页面上选择显示幻灯片内容时，则在上面所述的下拉菜单后面会多出一个  图标。单击这个图标，系统会弹出一个用于选择播放时间间隔倍增选项的弹出式菜单，如图 3-37 所示。

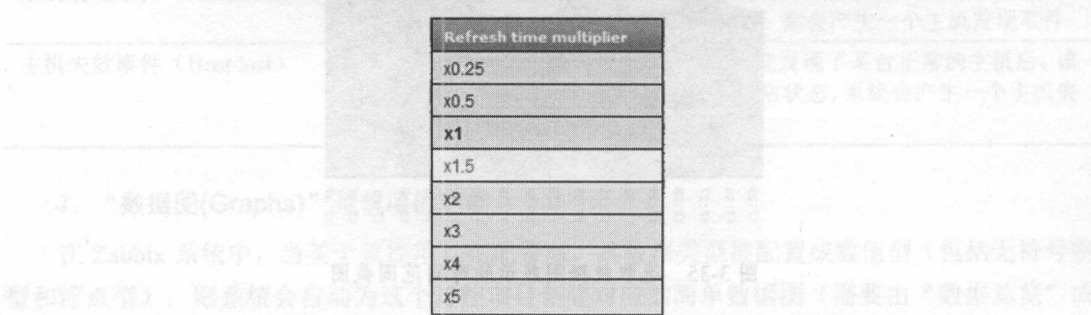


图 3-37 幻灯片页面弹出式菜单截图

这个菜单可以用来做什么呢？是这样的，如刚才我们所说的，所谓 Zabbix 幻灯片其实就是将多张图表页面组合在一起，然后系统会按照所设定的时间间隔自动地播放这些图表页面。播放两张相邻图表页面的时间间隔是我们在创建幻灯片时设置的，到播放时候就不能修改了。但是，如果想在播放幻灯片的时候调整幻灯片的播放快慢怎么办呢？这个时候就需要使用这个菜单了。当选择这个菜单上的不同菜单项时，对应幻灯片中的两张图表页面的播放延时，就调整为我们在创建幻灯片时所指定的延时值乘以所选择的菜单项的值。例如，如果在创建幻灯片时设置的延时值为 60 秒，而在这个菜单上选择“×2”菜单项，那么新的播放时间间隔就为 120 秒。

在图表页面上，还有一个需要说明的地方，那就是“组”和“主机”下拉菜单。如果创建并浏览过多个图表项目。那么你可能会留意到，图表页面上的“组”和“主机”下拉菜单，在某些图表页面会显示出来，而在另外一些图表页面却没有这样的下拉菜单。这是因为，当在创建图表时，如果该图表中有一个项目被你指定为“动态项目”，那么在浏览它的时候就会显示“组”和“主机”下拉菜单，否则，就不会显示这组下拉菜单。你或许还有一些迷惑，动态项目和非动态项目有什么差别啊？我们举个例子来说明，你可能就明白了。假如，有这么一组服务器，现在想要将它们 eth0 网卡的进出流量的简单数据图制作成图表，以用于动态展示使用。如果，我们手工一台一台将这组服务器中所有 eth0 网卡的进出流量图都添加到图表页面中，这样不但费时费力，而且还容易出错。在这种情况下，如果只将这组服务器中某台服务器的 eth0 网卡的进出流量图添加到图表中，然后在创建图表时选择上“动态项目”选项。此时，如果浏览这张图表页面，系统就会自动地将这组服务器中其他服务器的 eth0 网卡的进出流量图也添加到这个图表中，并且允许我们通过下拉菜单来选择显示谁的进出流量图。

9. “拓扑图(maps)”菜单项的功能

在 Zabbix 系统中，可以自己创建个性化的网络拓扑图。当在系统中创建了拓扑图后，通过“状态统计”→“拓扑图”页面，可以随时查看这些拓扑图信息。图 3-38 所示的就是我们自己定制的拓扑图。

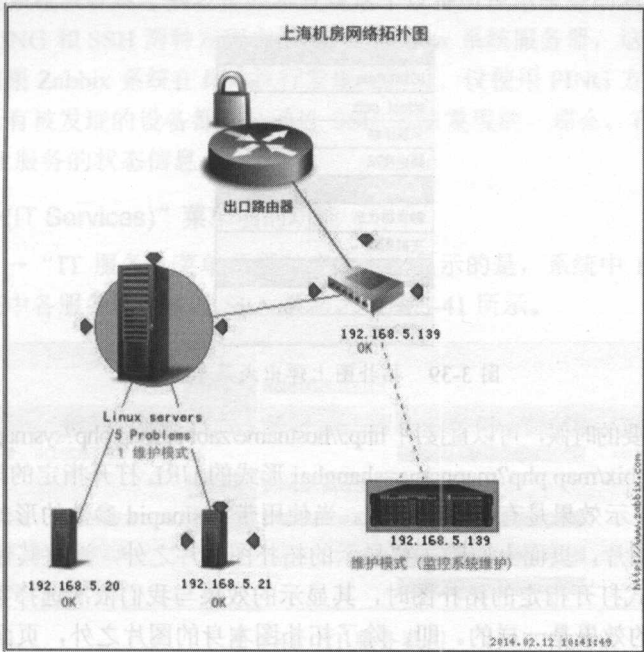


图 3-38 拓扑图截图

当拓扑图上的对象处于故障中时，该对象将会被一个填充了颜色的圆形图标加亮，如图 3-38 中所示 Linux Servers 主机组被一个红色圆形图标给加亮了，表示这个主机组中有主机出现了某种服务故障。而加亮圆形图的填充颜色与故障所对应触发器级别的颜色是一致的。如果所出现的故障（也就是触发器）都是已被确认的，则这个加亮圆形图标的圆周颜色是绿色的。

类似的，当拓扑图中某个主机处于维护状态时，则它将会被一个橙色矩形填充图加亮，如图 3-38 中所示的 192.168.5.139 这台服务器，就表示当前正处于维护模式下。当主机或其他对象处于被禁用状态时，则这个加亮的矩形填充图的填充颜色就为灰色。当然，上面所述的图标效果要想呈现出来，其前提条件是在创建拓扑图时，选中“图标高亮”选项。否则，就不会出现这些加亮效果。

在图 3-38 中可以看到“Linux Servers”、“192.168.5.139”、“192.168.5.21”等对象的周围都有 ◆ 图标，这表示该对象（或该对象所包含的主机）中有触发器的状态刚刚（触发器的状态改变时间不超过 30 分钟）改变过。类似的，要想在拓扑图中呈现上述的这种效果，则需要创建拓扑图时，选中 Mark elements on trigger status change 表单项。

在图 3-38 中，可以看出，连接不同对象之间的连接线有实线、虚线等。这些连接线的类型是我们在创建拓扑图时指定的。而且，实际上，在 Zabbix 系统的拓扑图中，连接线的类型不止这两种，并且可以根据主机中触发器所处的不同状态指定不同的连接线线型和颜色。关于如何指定对象之间连接线的类型和颜色，将在后续章节介绍如何创建拓扑图时详细介绍。现在只需要知道，在 Zabbix 系统拓扑图中，对象连接线的类型和颜色用户自己可以指定，并能根据触发器

的状态进行配置即可，以免具体介绍相关内容时产生疑惑。

当在拓扑图的某个对象上单击鼠标左键时，系统会弹出如图 3-39 所示的弹出式菜单。通过这个弹出式菜单，可以快速查看对应对象的触发器状态、主机图表（如果有的话）、执行指定的脚本以及打开一个指定 URL 连接的页面。这里的 URL 信息同样也是我们在创建拓扑图时配置的。



图 3-39 拓扑图上弹出式菜单截图

当在满足某些需要的时候，可以直接用 `http://hostname/zabbix/map.php?sysmapid=100100000000001` 或 `http://hostname/zabbix/map.php?mapname=shanghai` 形式的 URL 打开指定的拓扑图。用这两种形式打开拓扑图，其显示效果是有一定区别的。当使用带 `sysmapid` 参数的形式时，系统呈现给用户的仅仅是拓扑图图片，页面上，除了要显示的拓扑图图片之外，没有其他内容；而当使用带 `mapname` 参数的形式打开指定的拓扑图时，其显示的效果与我们依次选择菜单“状态统计”→“拓扑图”后显示的效果是一样的。即，除了拓扑图本身的图片之外，页面上还会显示菜单、导航条、状态栏等信息。而如果在打开指定拓扑图时，同时使用 `sysmapid` 和 `mapname` 两个参数，则参数 `mapname` 的优先级要高于 `sysmapid`。

10. “自动发现(Discovery)”菜单项的功能

“状态统计”→“自动发现”页面显示的是，通过网络自动发现功能所发现的主机信息列表，如图 3-40 所示。

自动发现规则			自动发现规则 全部
自动发现设备	已监控的主机	上线时间/下线时间	ICMP ping
ICMP Ping 192.168.5.x (118 设备)			
192.168.5.253	世纪二期19外网(C2960-24TT-L)	278 天, 18:21:26	
192.168.5.250	dell网络	309 天, 06:21:34	
192.168.5.135	-	615 天, 07:52:56	
192.168.5.20	-	17:47:26	
192.168.5.11	-	17:47:29	
192.168.5.10	-	263 天, 06:15:21	
			下线时间 17h 47m 23s

图 3-40 自动发现

单击“自动发现设备”列名，可以指定按所发现的设备名称顺序或倒序排列。如果被发现的主机已经被监控了，则“已监控的主机”列会显示出该主机被配置的主机名称。而“上线时间/下线时间”列显示的是，对应设备被发现或上一次被发现之后又被“丢失”了到现在的时长。

而最后一列显示的是，用于发现设备的服务当前的状态。当把鼠标指向对应的单元格时，系统会显示对应服务“在线”或者“下线”的时长。需要注意的是，在一个自动发现规则中，可以定义多个自动发现方法（实际上也就是某服务）。只有那些至少发现了一台设备的发现方法（或者称服务）才会在最后这一列里显示出来。换句话说，如果某些设备上开启了发现规则中对应的多个服务，但是系统只在这个列表里显示发现这个设备所使用服务的名称。例如，我们在发现规则中定义了 PING 和 SSH 两种发现方法，对于 Linux 系统服务器，这两种发现方法都可以发现它。但是，如果 Zabbix 系统在具体执行发现规则时，仅使用 PING 方法就发现了所有应该发现的设备，且所有被发现的设备都不是通过 SSH 方法发现的，那么，在图 3-40 所示的列表中不会显示 SSH 服务的状态信息。

11. “IT 服务(IT Services)”菜单项的功能

“状态统计”→“IT 服务”菜单项所对应的页面显示的是，系统中 IT 系统的状态情况。该页面显示了系统中各服务的状态及 SLA 数据，如图 3-41 所示。

root					
活动网站系统	严重问题	数据库端口状态不正常	-		
数据库服务群 - 数据库端口状态不正常	严重问题	数据库端口状态不正常	0.0456	99.9544 / 99.9999	
Web服务群 - HTTP状态不正常	OK	-	0.0184	99.9816 / 99.9900	
官网系统	严重问题	数据库端口状态不正常	-		
数据库服务群 - 数据库端口状态不正常	严重问题	数据库端口状态不正常	0.0456	99.9544 / 99.9900	
Web服务群 - HTTP状态不正常	OK	-	0.0184	99.9816 / 99.9900	
图片服务群 - HTTP状态不正常	OK	-	0.0045	99.9955 / 99.9900	

图 3-41 IT 服务截图

从图 3-41 中可以看出，“IT 服务”页面上显示的是各服务的状态及 SLA 数据列表。其中，如果服务的状态是“OK”，则表示该服务正常。而如果某个服务出现了问题，则状态列将以相关触发器所对应级别的颜色来显示该列中的信息。“Reason”列则显示的是，被触发的触发器的名称。而“Problem time”列显示的是 SLA 状态条，分别用红色和绿色表示对应服务的可用性和非可能性所占的比率。单击这个状态条上的超链接，系统将打开对应服务的可用性数据图页面，如图 3-42 所示。

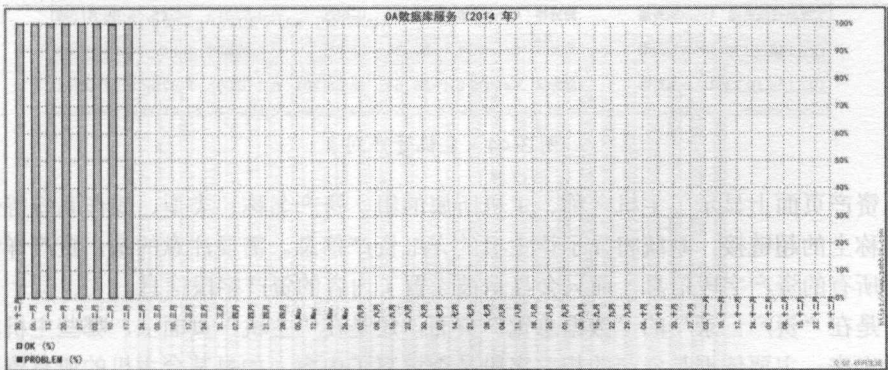


图 3-42 IT 服务可用性数据图

而“IT 服务”页面的最后一列“SLA/可接受的服务等级协议”则显示的是，当前 SLA 数据和对应服务可接受的最低 SLA 值。如果 SLA 当前值低于对应服务所配置的最低可接受值，

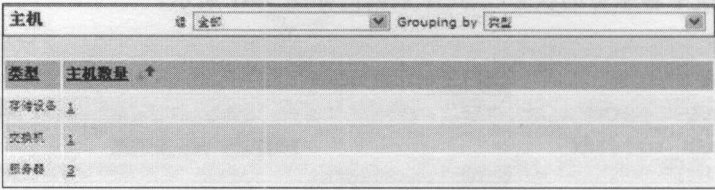
则对应服务当前 SLA 值用红色字体来显示，否则用绿色字体显示。单击服务名称上的超链接，系统将打开该服务的可用性报告页面。

3.3.2 “资产记录”菜单项的功能

“资产记录”主菜单项用于显示和浏览系统中所配置的硬件资源信息。这些硬件信息是我们在配置主机时，通过手工输入，或者将硬件资产信息的各个字段当作普通监控项目由 Zabbix 系统通过数据采集的方式采集的。“资产记录”主菜单项下有“数据总览”和“主机”两个子菜单项，它们的功能如下所述。

1. “数据总览(Overview)”菜单项的功能

“资产记录”主菜单项下的“数据总览”子菜单项所对应的页面，用来按照指定的资产字段汇总并显示系统中硬件资产信息，如图 3-43 所示。




主机	
全部 Grouping by 类型	
类型	主机数量
存储设备	1
交换机	1
服务器	2

图 3-43 资产记录数据总览页面截图

需要注意的是，数据总览页面在默认情况下可能不显示任何内容。需要从页面右上角的“组”和“Grouping by”下拉菜单里，选择适当的菜单项后页面才会显示相应的内容。当从“Grouping by”下拉菜单里选择“类型”这个菜单项后，系统将按照资产“类型”字段分组汇总系统中的资产信息，并将汇总后的资产信息显示在页面上，如图 3-43 所示。

2. “主机(Hosts)”菜单项的功能

在数据总览页面上单击“主机数量”列上的超链接，或者依次选择“资产记录”→“主机”菜单项，系统将显示系统中主机的资产信息列表，如图 3-44 所示。



主机	组	名称	类型	操作系统	编号1	标签	网卡物理地址1
天津监控服务器	Linux服务器		服务器	CentOS 5.7 x86_64	tj-653	YL-I-S2013	3C:E5:A6:B6:81:95
192.168.5.139	Linux服务器		服务器	CentOS 6.4 x86_64	nj-401	YL-I-S2008	D0:67:E5:EB:40:26
192.168.10.23	Linux服务器		服务器	CentOS 6.4 x86_64	Sh-nh-201	YL-I-S2006	F0:7B:CB:62:C0:F1

图 3-44 主机资产列表

主机资产页面上显示了主机名称、主机所属的组、资产名称、类型、操作系统等信息。单击主机名称上的超链接，系统将显示该主机的详细资产信息。需要注意的是，资产详情页面中不会显示所有的资产字段信息，而只会显示被设置了内容的资产字段信息。

不管是在“资产记录”的“数据总览”页面，还是在“主机”页面上，哪些主机的资产信息被显示出来，主要依据是资产的相关字段是否填写了内容。如果某台主机的所有资产字段都没有填写任何内容，则对应主机在“资产记录”主菜单项下的所有子菜单项所对应的页面上不会显示任何内容。从这一点上来讲，Zabbix 系统中的资产管理这一功能跟我们见到的其他资产管理系统还是有所不同的。

3.3.3 “系统报告”菜单项的功能

Zabbix 系统中的“系统报告”主菜单项是用来生成和显示系统统计报告信息的。在这个主菜单项下有 4 个子菜单项，它们分别是“Zabbix 状态”、“可用性统计”、“触发器前 100”和“统计报告”等。其中，“Zabbix 状态”子菜单项对应页面所显示的内容，与我们在前面介绍的“状态统计”→“状态面板”页面中“Zabbix 状态”数据区块中所显示的内容是一样的。所以，在此对该子菜单项所对应页面的功能就不再重述，如果读者有不明白的地方，可以参阅本书前面相关章节中的介绍。

而“可用性统计”子菜单项所对应页面显示的是，与所选定的主机或主机组相关的每个触发器，在指定的时间段内处于问题、正常（OK）和未知状态下的时间统计及占总时间的比例。通过这个功能页面，可以很方便地查看某台主机，在指定的时间段内分别都是因为什么原因造成了该主机或该主机上某些服务不可用，以及它们导致主机或其上服务不可用所占的时间与总时间的比例，从而找出导致主机或其上服务不可用问题的主要原因，如图 3-45 所示。

名称	Problems	Ok	未知	数据图
5672端口连接失败不通	2.7551%	97.2449%	0.0000%	数据图
6388端口连接失败不通	2.4221%	97.5779%	0.0000%	数据图
8080端口连接失败不通	2.4221%	97.5779%	0.0000%	数据图
CPUI占用率持续6小时大于90% #763	0.0000%	97.2654%	2.7346%	数据图
CPUI占用率持续6小时大于90% #763	0.0000%	97.2641%	2.7359%	数据图
CPUI占用率持续6小时大于90% #770	0.0000%	97.2628%	2.7372%	数据图
CPUI占用率持续6小时大于90% #771	0.0000%	97.3077%	2.6923%	数据图
CPUI占用率持续6小时大于90% #772	0.0000%	97.5270%	2.4730%	数据图
CPUI占用率持续6小时大于90% #773	0.0000%	97.5259%	2.4741%	数据图
CPUI占用率持续6小时大于90% #774	0.0000%	97.5247%	2.4753%	数据图
CPUI占用率持续6小时大于90% #775	0.0000%	97.5235%	2.4765%	数据图
WebUI占用率持续6小时大于90% #776	2.7549%	97.2451%	0.0000%	数据图

图 3-45 主机和服务可用性统计列表

在如图 3-45 所示的主机和服务器可用性统计页面上，单击触发器名称上的超链接，系统将会打开对应触发器最新事件的页面。而单击“数据图”列上的“显示”超链接，系统将会显示对应监控项目的可用性柱状图，该柱状图的每一条柱形都代表当前年份内，一个星期时间段内对应监控项目的可用性情况。其中，红色柱形代表主机或服务处于不可用状态下的时间占总统计时间的百分比；绿色柱形则代表主机或服务处于可用状态下的时间占总统计时间的百分比；而如果柱形是灰色的，则代表该段时间内没有统计数据，如图 3-46 所示。

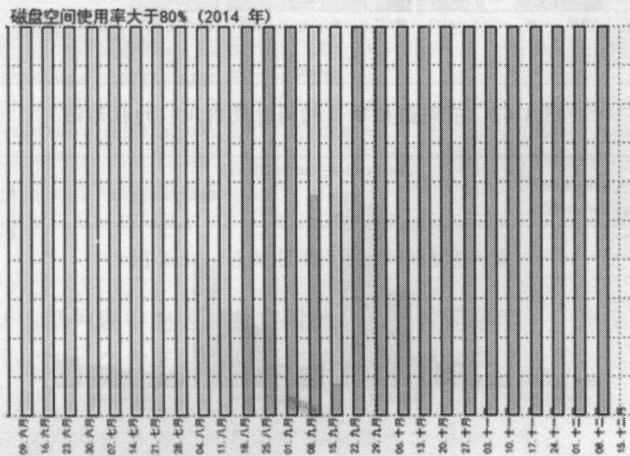


图 3-46 主机或服务可用性柱状统计图

“系统报告”主菜单项下的“触发器前 100”子菜单项所对应页面显示的内容比较简单，从这个子菜单项的名称就可以看出，该子菜单项所对应页面显示的是，系统中在指定的时间段内，触发器状态改变次数最多的前 100 个触发器的信息。因此，这个子菜单项所对应的页面功能，在这里就不再赘述了，请读者自行研究。

“系统报告”主菜单项下的最后一个子菜单项“统计报告”子菜单项，在 Zabbix 系统的官方手册中被称之为“Bar Reports”。它所对应的页面，是用来创建并显示个性化的柱状图的。需要特别说明的是，通过该子菜单项对应的页面，所创建的柱状统计报告图只能当时查看，无法保存。

在创建统计报告时，需要用户输入一些表单项的值。在这些需要输入值的表单项中，像“标题”、“X 轴标签 (X label)”、“Y 轴标签 (Y label)”、“是否显示图例 Legend”、“周期 (Period)”、“项目 (Item)”、“主机组”、“主机”、“调色板 (Palette)”等表单项的含义和作用，读者一看表单项名称即可理解，即使有一时不清楚的，只要测试一次也就很快清楚了。所以，对于这些表单项的含义和作用，我们在这里不再一一作出说明。

接下来，需要做出简单说明的表单是“比例 (Scale)”。对于“比例 (Scale)”表单项，笔者个人认为，其名称与它的实际作用和含义并不完全吻合。实际上，它设定的是统计的分隔周期。即，是指按小时、天、星期、月还是按年分别进行统计，并生成报告。所以，依照笔者的理解，这个表单项取名为“统计区间”或“统计间隔”或许更贴切一些。

在“统计报告”页面的右上角有一个“系统报告”下拉菜单，通过这个下拉菜单，可以选择所要创建的报告的类型。这个下拉菜单总共有 3 个菜单项，它们分别是 Distribution of values for multiple periods，表示创建不同的监控项目在相同的时间区间内比较的柱形图，如图 3-47 所示；如果选择 Distribution of values for multiple items 菜单项，则表示创建同一个监控项目在不同的时间区间内比较的柱形图，如图 3-48 所示；最后一个菜单项 Compare values for multiple periods，表示创建多个主机组或主机中某类指定的监控项目（所选定的主机或主机组均需要有这类监控项目）在指定的时段内数据比较的柱形图，如图的 3-49 所示。

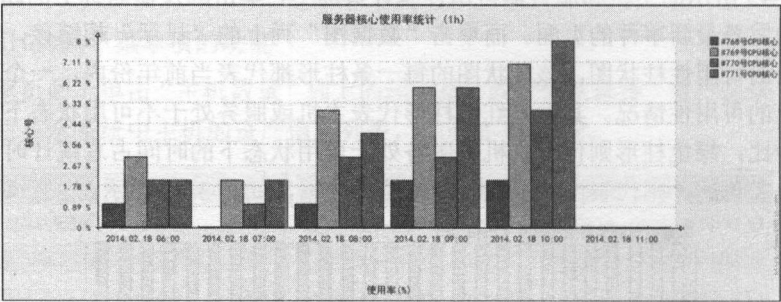


图 3-47 不同监控项目相同时段内比较的柱形图

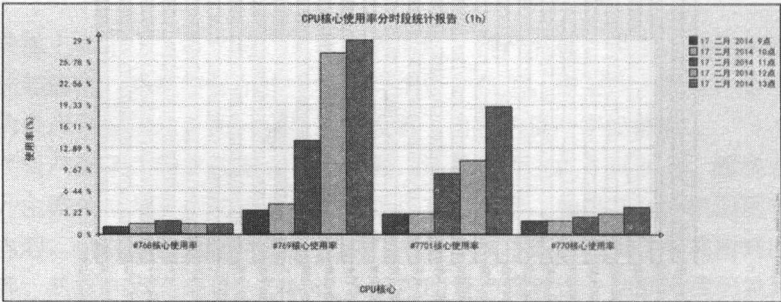


图 3-48 同一个监控项目在不同时段比较的柱形图

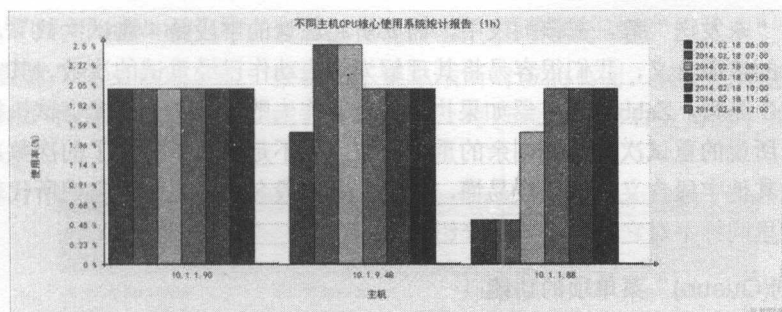


图 3-49 不同主机的同一类监控项目比较的柱形图

3.3.4 “高级配置”菜单项的功能

“高级配置”主菜单项是用来管理 Zabbix 系统的，只有超级管理员用户才有权限使用该菜单项及其下的各子菜单项的功能。所以，如果通过 Zabbix 系统 Web 前端页面无法看到这个主菜单项，则需要确认你所登录的用户是否属于超级管理员组。“高级配置”主菜单项下包括“常规 (General)”、“节点管理 (DM)”、“认证方式 (Authentication)”、“用户 (Users)”、“消息介质 (Media types)”、“脚本 (Scripts)”、“审计 (Audit)”、“队列 (Queue)”、“报警 (Notifications)”和“安装 (Installation)”等子菜单项。本节我们着重讨论“审计”、“队列”、“报警”和“安装”等子菜单项的功能，其他子菜单的功能将在后续章节中陆续进行说明和介绍。

1. “审计(Audit)”菜单项的功能

“高级配置”主菜单项下的“审计”子菜单项所对应的页面，主要是用来显示用户通过 Zabbix 系统 Web 前端组件对 Zabbix 系统所进行修改的记录信息，以及系统中所配置的“动作”被执行情况的详细信息。默认情况下，该页面显示的是用户对 Zabbix 系统所做修改的记录信息（对应于“日志”菜单项），通过页面右边上部的下拉菜单，可以选择显示“动作”执行情况的详细信息（对应于“动作”菜单项），如图 3-50 所示。

Zoom: 1h 2h 3h 6h 12h 1d 1w 2w 1m 3m 6m 1y 全部							
< >							
<< 1y 5m 1m 1w 1d 12h 1h 1h 12h 1d 1w 1m 6m 1y >>							
时间	用户	IP地址	资源	动作	ID	描述	详细
11 九月 22:33:48	Admin	180.157.211.248	用户	登录	1		
11 九月 21:43:03	Admin	180.157.211.248	用户	已更新	0		User alias [Admin] Name [Zabbix] Surname [Administrator] profile id [1]
31 八月 12:27:58	Admin	180.157.211.248	用户组	启用	0		Debug mode for group name [Zabbix administrators]

图 3-50 系统操作日志审计信息列表

“日志”型审计信息列表中绝大部分字段的含义比较简单明了，相信读者一看各字段的名称就能看明白各字段代表什么含义。所以，在这里就不对这些字段的含义一一说明了。仅需要简单说明的是“ID”字段中内容的含义，它所代表的是被操作对象在 Zabbix 系统数据库中的 ID 编码。如果这个字段内的内容是 0，则表示对应对象并非单个对象。

“动作”型的审计信息列表中，“状态”字段内的内容可能是“正在进行”、“已发送”、

“已执行”和“未发送”等。在该列表中，需要引起注意的字段是“重试次数”。这个字段，从其中文名称的字面含义，我们很容易将其理解为对应动作已经重试的次数。其实不然，这里的“重试次数”是指，Zabbix 服务器如果执行对应动作失败，则还会重新尝试执行多少次。也就是说，这里所说的重试次数是指剩余的重试次数，而不是已经重试过了的次数。该类型审计信息列表中的其他字段含义同样比较易懂，读者一看字段名称即可知道它们所代表的含义。所以，我们在这里同样不对它们做一一的介绍和说明。

2. “队列(Queue)”菜单项的功能

“高级配置”主菜单项下的“队列”子菜单项所对应的页面，是用来查看 Zabbix 系统中监控项目被延时采集监控数据的队列信息。

理想情况下，如果打开这个页面，页面上所有数据行都是绿色的，则这意味着队列中没有被延时采集监控数据的监控项目。在这种情况下，同样也意味着，所有监控项目都没有被延迟采集监控数据，即所有的监控项目都是按照配置计划采集监控数据，并更新到系统数据库中的。然而，有时候可能因为 Zabbix 服务器的性能问题、网络连接问题、被监控设备代理的问题，等等，会有一些被监控项目的监控数据被延时采集，或者被延时更新到数据库中。此时，我们通过“队列”页面就可以查看这些被延时采集监控数据的监控项目的信息，如图 3-51 所示。

已被更新的项目队列						
数据总览						
项目	5秒	10秒	30秒	1分	5分	10分钟以上
Zabbix agent	0	0	0	1	0	0
Zabbix agent(主动方式)	0	0	0	0	0	0
Simple check	0	0	0	0	0	0
SNMPv1 agent	0	0	0	0	0	0
SNMPv2 agent	0	9	157	146	0	1
SNMPv3 agent	0	0	0	0	0	0
Zabbix internal	0	0	18	2	0	0

图 3-51 队列列表

Zabbix 系统中的队列，仅表示监控数据是否被及时更新的逻辑状态，而不像其他很多软件中那样，队列涉及进程间通信，具有先进先出的机制。实际上，Zabbix 系统中所谓的队列并不是真正意义上的队列，它更像是监控项目的监控数据被延时了不同时长的统计集合。

通过“队列”页面右上角的下拉菜单，可以在三种不同统计格式中选择一种，来统计和显示系统中的队列信息。这三种统计格式分别是：“数据总览”，按照监控数据采集方法的不同，分组汇总出系统中队列信息；“Overview by proxy”，按照分布式节点分组汇总系统中队列信息；最后一项“详细”，则详细列出系统中被延时更新的监控项目及它们所对应的主机、被延时的时间等详细信息。

队列页面中的 5s、10 s、30 s 各字段表示，被延时了 1~5 s、5~10 s、10~30 s 采集监控数据的监控项目的数量。该页面中其他字段的含义比较简单，相信读者根据字段名称就可以看出它们的含义，在此就不一一叙述它们的作用了。

在前面提到过，如果监控服务器的性能有瓶颈，则可能会反映在队列页面的统计结果上。所以，参考队列页面的统计结果，也是我们在对 Zabbix 系统进行调优时确定 Zabbix 服务器是否遇到性能瓶颈的一个方法。

3. “报警(Notifications)”和“安装(Installation)”菜单项的功能

“高级配置”主菜单项下的“报警”子菜单项所对应的页面，显示的是指定的时间段内发送给每位用户报警信息的汇总。统计的时间段，可以通过页面右上角的下拉菜单来选择，如图 3-52 所示。

报警			
消息介质类型 全部 周期 每周 年 2014			
起始	截止	Admin	Guest
30 十二月 2013 00:00	06 一月 2014 00:00	0 (0/0/0/0/0)	0 (0/0/0/0/0)
06 一月 2014 00:00	13 一月 2014 00:00	0 (0/0/0/0/0)	0 (0/0/0/0/0)
13 一月 2014 00:00	20 一月 2014 00:00	0 (0/0/0/0/0)	0 (0/0/0/0/0)
20 一月 2014 00:00	27 一月 2014 00:00	52 (0/1/33/19/0)	0 (0/0/0/0/0)
27 一月 2014 00:00	03 二月 2014 00:00	0 (0/0/0/0/0)	0 (0/0/0/0/0)
03 二月 2014 00:00	10 二月 2014 00:00	0 (0/0/0/0/0)	0 (0/0/0/0/0)

图 3-52 系统报警信息统计

如图 3-52 所示，报警统计信息页面的每列显示了一个用户在某个时间段内被发送报警信息的条数。统计数据的格式为：总数(通过消息介质 1 发送的总数/通过消息介质 2 发送的总数……)。

“高级配置”主菜单项下的“安装”子菜单项的功能很简单，通过它，可以启动 Zabbix 系统 Web 前端组件重新部署的安装向导。Zabbix 系统 Web 前端组件部署的安装向导，我们在前面章节中已经介绍过如何使用它了，在此不再复述。但是，需要注意的是，如果单击了这个子菜单项的超链接，而又在中途取消重新部署 Zabbix 系统 Web 前端组件，则不能简单地关闭相应浏览器窗口了事，而是需要单击安装向导页面上的“取消”按钮来取消本次部署操作。否则，即使关闭了打开安装向导的浏览器窗口，但在之后重新打开 Web 页面时，Zabbix 系统仍然显示安装向导中被中断的步骤所对应的页面。

3.4 配置第一台被监控主机

前面提到过，在 Zabbix 系统中，所有可以被监控的网络设备，包括服务器、交换机、路由器、防火墙、存储设备、各类虚拟机都称之为被监控主机。总之，在监控网络中，可以被独立配置网络 IP 地址或主机名的，且可以被监控的各类物理或逻辑设备，都被称为被监控主机。在后面的叙述中，我们有时候也将被监控主机直接简称为主机或被监控主机，读者只需将主机、被监控主机视为同一个概念即可。

或许你在阅读本书的前面章节时，早就急不可耐地想在 Zabbix 系统中添加被监控主机了吧。下面就来介绍一下如何通过 Web 前端页面在 Zabbix 系统中添加第一台主机。其实，我们称“第一台主机”是不确切的。因为，当依次选择“系统配置”→“主机”菜单后，可以从主机列表中看到，系统已经预定义了一台被称之为“Zabbix 服务器”的主机。这台叫“Zabbix 服务器”的主机，其实就是我们所安装的 Zabbix 服务器，其是在我们完成 Zabbix 系统安装部署后，由 Zabbix 系统自动为我们添加上的。所以，这里所说的第一台被监控主机是指，我们自己主动通过手工向 Zabbix 系统中添加的第一台被监控主机。

在 Zabbix 系统中添加一台新的主机其实非常简单，只需通过 Web 页面，依次选择菜单“系统配置”→“主机”，在主机列表页面上，单击页面右上角的“新建主机”按钮，系统将打开创建新主机的表单，如图 3-53 所示。

主机名

显示名称

组

已加入组

Linux服务器

其他组

Discovered hosts

Linux servers

Templates

Zabbix servers

新建主机组

Agent接口

IP地址

127.0.0.1

主机域名

连接到

IP地址

主机域名

端口

10050

默认

添加

SNMP接口

添加

JMX接口

添加

IPMI接口

添加

由代理节点监控

(不通过代理节点)

状态

检测中

图 3-53 创建新主机表单

从图 3-53 所示的截图中可以看出，在创建新主机表单页面上，系统提供了“主机”、“模板”、“IPMI”、“宏变量”以及“主机资产”等选项卡。而每个选项卡，系统又为我们提供了很多表单项。在这些众多的表单项中，通过前面的介绍，你大概知道了其中一部分表单项的作用；而还有一些表单项的作用和含义则一时还弄不清楚。表 3-5 列出了“主机”选项卡上各表单项的含义和作用。

表 3-5 “主机”选项卡表单项列表

表单项	描述
主机名 (Host name)	这个表单项要求输入的是Zabbix系统中唯一的主机名。主机名可以由字母、数字、空格、点号和下划线等字符组成。经我们测试，主机名也可以支持中文字符。但是需要注意的是，如果被配置的主机上，运行了被监控设备代理进程（agent），则被监控设备代理进程的配置文件中，Hostname配置项所配置的内容，需要与这里输入的内容保持一致。因为，在主动模式下，Zabbix服务器端（或Zabbix服务器代理端）进程会依据Agent提供的主机名，在系统中查对主机名。而Agent进程所提供的主机名，即为它的配置文件中Hostname配置项所指定的内容。该表单项为必填项
显示名称 (Visible name)	这个表单项的含义如它的名称所揭示的那样，在配置一台主机时，如果这个表单项被输入了内容，那么在显示主机列表或者将该主机应用到拓扑图上时，则对应主机的名称即为该表单项所输入的内容。该表单项为可选项，如果该表单项不输入任何内容，则在要显示主机名称的地方，系统将会显示“主机名”表单项里所输入的内容

续表

表单项	描述
组 (Groups)	这个表单项的含义比较好理解，即选择被创建主机所属于的主机组。一台主机至少要属于一个主机组
新建主机组 (New host group)	如果这个表单项里被输入了内容，则在创建主机的同时，系统会自动创建一个新的主机组，同时将新创建的主机添加到这个新创建的主机组中。新创建的主机组的组名即为这个表单项里所输入的内容。如果这个表单项为空，则在创建新主机时，忽略该表单项
接口 (Interfaces)	在添加主机的表单页面上，没有一个叫做“接口”的表单项。我们这里所称的“接口”，是对该表单页面上各种类型接口表单项的总称，它包括Agent接口、SNMP接口、JMX接口和IPMI接口。在前面介绍过Zabbix系统所支持的各种监控数据采集方法，这几种接口类型就对应各自的监控数据采集方法。“IP地址”和“主机域名”表单项一般就填写被监控主机的IP地址或它的合法DNS域名。“IP地址”/“主机域名”按钮，则是让用户选择Zabbix服务器是使用IP地址的形式连接被监控主机，还是使用主机域名的形式连接被监控主机，两者二选一。单击接口表单项数据块里的“添加”超链接，可以添加多个接口。在Zabbix系统中，同一台主机，即使是同一类型的接口，也可以被添加多个。而单击“移除”超链接，则可以删除相应的接口。但是，需要注意的是：如果有监控项目被配置成使用了某个接口，则对应接口后面的“移除”超链接将会变灰色且不能被单击。需要将所有使用了该接口的监控项目都删除之后，对应的接口才可以被删除。一台被监控主机至少需要配置一个接口，否则在添加新主机时系统会报错
端口 (Port)	即为对应接口的端口号。如果接口类型是Zabbix被监控设备代理，则默认值为10050；SNMP类型接口的默认端口号是161；IPMI类型接口的默认端口号为623
默认 (Default)	在每种类型接口表单项里都有一组“默认”单选按钮。当针对某种类型配置了多个接口时，则需要选择一个默认接口。这个默认接口是做什么用的呢？当在为某台主机配置监控项目时，需要针对所配置的监控项目配置相应的接口，难道配置默认接口就是为了在配置监控项目时能让默认选项显示在“主机接口”表单项上？经过我们实际测试，实际情况不是这样的。不管我们选择哪个接口作为默认接口，在“主机接口”表单项上所显示的默认选项仍然是按顺序显示的。不知读者是否还记得，我们在前面章节中介绍过，在Zabbix系统中有“模板”的概念。在模板里是可以配置监控项目的，但是配置在模板里的监控项目中是没有指定接口的。所以，当把一个模板关联到一台主机时，如果某个监控项目所使用到的某种类型的接口，在该台主机上被配置了多个，则来自模板中的监控项目将会被自动关联到默认接口上。所以，这里我们所配置的默认接口就是应用到这个场景下的
由代理节点监控 (Monitored by proxy)	在前面介绍Zabbix系统的组件时，介绍过在完整的Zabbix系统中，有一个叫做服务器代理 (Proxy) 的组件。它的作用就是代替Zabbix服务器采集被监控主机的监控数据，并将所采集到的数据定期发送给Zabbix服务器，由Zabbix服务器来处理这些数据。这里的这个表单项就是用来配置将要添加的主机是否由服务器代理来采集监控数据，以及由哪台服务器代理（当配置了多台服务器代理时）来采集监控数据。可以选择“不通过代理节点”选项，即表示新添加的主机不通过任何服务器采集监控数据，而由Zabbix服务器自己采集这些数据
状态 (Status)	这个表单项应该很好理解。当选择“检测中”这个选项时，则表示正常启用对新添加主机的检测与监控；而当选择“不检测”选项时，则表示禁用该主机，即表示不对该主机采集监控数据与报警

在表 3-5 中列出了“添加主机”页面上“主机”选项卡中，添加主机表单里各个表单项的含义和作用，或许，你还是没有完全搞懂如何去填写它们。没有关系，我们在这里主要是想介绍一下，在 Zabbix 系统中添加主机的一般过程和方法。所以，你并不需要在上述表单中的每个表单项里都填写内容，而只需填写“主机名”、“主机组”和“Agent 接口”中的 IP 地址，而其他表单项保持默认值即可。完成上述内容的填写后，单击表单下部的“保存”按钮，以保存我们所配置的信息。这样，一台被监控主机就被添加到 Zabbix 系统中了。

但是，为了使我们在后面所介绍的测试能够顺利进行，请读者自行按照我们在前面章节中介绍的方法，在 Agent 接口表单项指定的 IP 地址所对应的主机上安装配置好被监控设备代理组件（即 agent），并启动它。

3.5 配置监控项目

在 Zabbix 系统中，监控项目是所有被采集的监控数据的载体，没有监控项目就无从谈起监控数据的采集。这是因为，我们只有在监控项目上才能定义和配置监控数据所采用的采集方法、数据类型、数据的单位以及所采集的数据所属于的主机。即使数据采集方法使用我们在前面介绍的“数据聚合”和“通过计算的方法采集数据”，它们也是需要依附于监控项目的。一个主机所拥有的监控项目主要是通过两种方法创建的：第一种方法即是我们将要介绍的，针对某台特定主机，通过手工创建；第二种方法是，将一个模板关联到主机，则该模板上创建的监控项目就会被自动关联到对应主机上，并在对应主机上自动生成相应的监控项目。通过这两种方法创建监控项目各有优缺点，手工方式创建监控项目比较灵活、方便，可以适用于所有情况下监控项目的创建。但是，如果被监控主机和监控项目的数量比较多，那么通过这种方式创建监控项目工作量就很大，很难进行大批量的添加，容易出错且不易管理。反之，通过模板关联的方式创建监控项目则具有高效、方便日后管理等优点。但是，通过这种方法添加监控项目，灵活性较差，当被监控主机的类型比较多时，维护模板会增加我们一定的工作量。

因为任何一个监控项目都要依附于某台主机，所以，要创建新的监控项目，就需要依次选择菜单项“系统配置”→“主机”，打开主机列表页面，并从这个主机列表页面上找到我们所要添加监控项目的主机。因为我们还没有向刚刚添加的主机上添加任何监控项目，所以，从主机列表页面上，可以看到刚刚添加的主机的“项目”为 0 个。单击“项目”上的超链接，系统会打开对应主机已经被配置的监控项目的详细信息列表。单击列表页面右上角的“新建项目”按钮，系统会打开添加监控项目的表单页面，如图 3-54 所示。

在添加监控项目的表单上有很多表单项，它们的作用和含义如表 3-6 所示。图 3-54 给我们最直观的感觉就是，如果要添加一个监控项目，则我们需要输入和填写的内容还是比较多的。这些表单项，有些我们一看它们的名称就知道它们的作用和含义，而另外一些则需要我们做进一步的测试，才能对它们的作用和含义有较深的理解。这里我们主要是想介绍一下添加一个监控项目的一般方法和过程，所以你只需按照所述的，在这个表单页上填写以下表单项内容即可：“名称”表单项里填写“CPU 负载”，当将这个监控项目配置完成之后，在其他页面里显示这个监控项目的信息时，系统所使用的名称即是在“名称”表单项里所输入的“CPU 负载”；“关键字”表单项里输入“system.cpu.load”；“信息类型”表单项选择“数值（浮点数）”选项。然后，单击页面下部的“保存”按钮，以保存对该监控项目的配置。

主机

名称

类型

Key

主机接口

信息类型

单位

乘自定义值

更新间隔(秒)

Flexible intervals

No flexible intervals defined.

New flexible interval

保留历史数据(天)

保留趋势数据(天)

取值方式

显示值

新建监控分类

选择

127.0.0.1 : 10050

数值(浮点数)

1

30

间隔

周期

动作

间隔(秒)

周期

添加

90

365

原始数据

原始数据

显示数据映射

图 3-54 添加监控项目表单截图

提示：为了减小 Zabbix 系统数据库的数据量，进而减小数据库的压力，建议你在配置监控项目时将“保留历史数据（天）”表单项的数值调小，例如将其调整为 7 天或 15 天等。

如果新添加的监控项目所在的主机上已经安装配置好了 Zabbix 被监控代理组件，并开启了该组件对应的进程，则完成监控项目的添加后，约等 1~2 分钟，就可以通过我们在前面章节中介绍的方法，依次选择“状态统计”→“最新数据”菜单项，并在“组”和“主机”下拉菜单中找到对应主机，就可以查看你刚刚添加的监控项目有没有正常地采集到监控数据。如果所添加的监控项目没有正常采集到监控数据，则需要仔细检查是否是哪一步配置不正确。

表 3-6 添加监控项目表单项列表

表单项	描述
主机(Host)	显示被添加监控项目所属于的主机。该表单项不可直接输入，但是可以通过其后的“选择”按钮选择不同的主机
名称(Name)	即为被添加监控项目的名称。监控项目的名称除了支持普通的字符文本外，还可以支持\$1、\$2、\$3、……、\$9等宏变量。而这些宏变量代表的是，对应监控项目关键字中的第几个参数。例如，如果监控项目的关键字为vfs.fs.size[/,free]，则\$1就代表“/”，而\$2则代表“free”。当监控项目名称使用宏变量时，宏变量会在该监控项目的名称被使用时（例如在页面上显示或使用在报警信息中），被所对应的值替换掉
类型（Type）	这个表单项中的选项，与我们在前面章节中所介绍的，监控数据采集方法是相对应的
关键字（Key）	关键字表单项既可以输入又可以选择。当在“类型”表单项中选择某些类型时，例如Zabbix Agent（包括主动方式）、Zabbix internal、数据聚合（Zabbix aggregate）等类型时，则它们对应监控项目关键字的格式都是相对固定的，由Zabbix系统内部程序确定。所以，此时应尽量通过选择的方式确定关键字。而当选择另外一些类型的时候，则关键字可以自由指定，只要指定的关键字符合Zabbix系统关于关键字的命名规范即可。需要说明的是，属于

续表

表单项	描述
	同一台主机上的所有监控项目的关键字都必须是唯一的，不能有重复。但是，不同主机上监控项目的关键字则是可以重复的。另外，在Zabbix系统中，监控项目的关键字比较重要，当在系统内部引用一个监控项目时，往往需要使用到该监控项目的关键字。例如，在前面介绍的“数据聚合”和“通过计算的方法采集监控数据”等监控数据采集方法中，以及在后面将要介绍的触发器中，都需要引用监控项目的关键字。关于监控项目关键字的命名规范，以及在关键字中如何使用参数等内容，我们将在后续章节中作详细的介绍
主机接口 (Host interface)	在前面介绍如何创建主机时，介绍过“接口”的概念。这里是指，从我们之前在添加主机时所定义的接口中选择一项，用于当前监控项目监控数据的采集。另外，有一些监控数据采集方法并不需要接口。例如，数据聚合、通过计算的方法采集监控数据、Zabbix系统内部数据采集等等，当将新添加的监控项目的类型选择为这些类型的时候，系统将隐藏掉这个表单项
消息类型 (Type of information)	该表单项指定，将新添加的监控项目所采集到的监控数据，以何种数据类型格式存储在Zabbix系统数据库中。当前系统支持的信息类型有：数值（无符号整型）[Numeric (unsigned)]，即表示64位的不带符号的整数；数值（浮点数）[Numeric(float)]，表示浮点数，可以允许是负数，支持的数值范围为-999 999 999 999.999 9到999 999 999 999.999 9；字符串（Character）型，即表示字符串型的数据，最大字符串的长度为255个字符；日志型，表示日志类型数据，如果监控项目的关键字，被配置成log[]形式的，则它的信息类型必须要使用日志型；文本型，即表示文本形式的数据，其长度可不受限制
数据类型（Data type）	数据类型是针对消息类型为整型的监控项目而言的。换句话说，如果在“消息类型”中选择的不是“数据（无符号整型）”，则“数据类型”表单项将不起作用。实际上，如果在添加或配置监控项目时，在“消息类型”表单项中选择的不是“数据（无符号整型）”选项时，则系统会自动隐藏该表单项。“数据类型”表单项用于指定整型数据属于哪种格式的数据。这些数据格式包括：布尔型（Boolean），即表示将数据文本转换成表示逻辑“真”与“假”状态的0和1。需要说明的是，布尔型监控项目不仅可以接受0和1这两个表示“真”和“假”的数据。而且，当布尔型的监控项目接收到true、t、yes、y、on、up、running、enabled和available等字符串时，系统都将其视为合法的布尔型数据，即将其视为逻辑“真”，并将其转化为数字“1”存入到Zabbix系统数据库中；相应的，当系统接收到以下布尔型的数据时，系统将会将其转化为数字“0”，并将其存入到Zabbix系统数据库中：false、f、no、n、off、down、unused、disabled及unavailable等；另外，当系统接收到非零的整型数据时，则系统将其视为逻辑“真”看待；而当系统接收到数字0时，系统将其视为逻辑“否”。另外几种数据类型是：“八进制（Octal）”，表示对应的数据是八进制数据；“十进制（Decimal）”，表示对应的数据是十进制数据；“十六进制（Hexadecimal）”，表示对应的数据是十六进制数据
单位（Units）	如果在添加监控项目时配置了单位，则Zabbix系统在接收到采集的监控数据后，会进行后续的加工处理。并且在显示对应数据时，系统会将配置的单位信息与数据一起显示在页面上（直接将单位附加在数据的后面）。但是，在以下情况，系统将会做一些特殊处理。第一，默认情况下，如果系统接收到的数据值超过1000，则系统会将原始数据除以1000，并且会在单位符号前面添加上递级的单位符号。例如，如果监控项目配置的单位符号为温度符号℃，那么当系统接收到像1004.2这样超过1000的数据时，会自

续表

表单项	描述
	<p>动将其转化为1K℃这样的数据，而不管实际生活中是否存在“K℃”这样的单位符号。类似的，当系统接收到的数据值超过1 000 000时，则系统会自动在单位符号前面添加上M。第二，如果该表单项所设置的单位符号为B（字节）或B/s（字节每秒）时，则单位递级的除数为1024，而不是1000。即，如果系统接收到的数据为1536B/s时，则系统会自动将其转化为1.5KB/s。第三，当监控数据的单位被设置成“unixtime”时，则系统会自动将所接收到的“整型（无符号整型）”数据转化为“yyyy.mm.dd hh:mm:ss”格式的时间格式数据。第四，当监控数据的单位被设置为“uptime”时，则系统会自动将所接收到的数据转化成“hh:mm:ss”或“N 天，hh:mm:ss”格式的时间数据。例如，当系统接收到的数据为881 764时，则该数据将会被转换成“10天，04:56:04”的格式。第五，当数据的单位被配置成“s”时，则系统只取转化后数据的前3部分数据。即，当系统接收到的数据为大于一天的秒数（86 400）时，则系统会将其转换为“n天m小时k分钟”形式的数据。例如，如果系统接收到的数据为87 040，则系统会将其转化为1d 10m，中间的0小时省略了未显示。而如果系统所接收到的数据为小于86 400而大于1时，则系统会将其转换成“小时分钟秒”形式的数据。例如，如果系统接收的数据为86300，则系统会将其转化为“23h 58m 20s”。而当系统所接收的数据小于1时，则系统将其转化为毫秒格式的数据。例如，如果系统接收到的数据为0.3，则系统自动将0.3转化为300ms格式。需要特别说明的是，我们在这里所说的数据格式转换仅发生在数据显示时，在Zabbix系统数据库中存储的仍然是原始数据</p>
乘自定义值 (Use custom multiplier)	<p>配置了这个表单项，系统会自动将所接收到的整型或浮点型数据乘以这个表单项所配置的乘数，并将所得的结果存入到Zabbix系统数据库中。如果系统接收到的原始数据本身就是KB或GB格式的数据，使用这个表单项可以将系统所接收到的数据还原成字节格式，以免系统在自动将其计算成KB或GB格式的数据时，计算结果不正确。需要指出的是，这个表单项与上面所介绍的“单位”表单项不同。如果在“乘自定义值”表单项里设置了值，则系统数据库中存储的是原始数据乘以乘数后的结果，而不是原始数据</p>
更新间隔 (Update interval, 单位是秒)	<p>配置监控项目所对应监控数据的采集时间间隔，单位是秒。如果该表单项填写的是0，则对应监控项目将不会被采集监控数据</p>
可变更更新间隔 (Flexible intervals)	<p>上面介绍的“更新间隔”表单项用于配置对应监控项目默认的监控数据采集时间间隔。除了默认的监控数据采集时间间隔，在Zabbix系统中，针对同一个监控项目，还可以配置不同的时段采用不同的监控数据采集时间间隔，这就是可变更更新间隔。例如，可以配置可变更更新间隔为10s，周期为1-5,9:00-18:00，表示该监控项目在每个星期的星期一到星期五的上午9点到下午6点这些时间区间内，采用10s的时间间隔采集监控数据，其他的时间段则使用默认的时间间隔采集监控数据。每个监控项目最多可以配置7条可变更更新间隔规则。在这些可变更更新间隔规则中，如果所指定的时间段有重叠，则在重叠的时间段内，系统将采用规则中最小的更新间隔作为实际监控数据采集间隔。如果默认更新间隔被设置为0，而可变更更新间隔被配置成非0的数字，则系统将依照可变更更新间隔中指定的时间间隔进行监控数据的采集；相反，如果在某个时间段内，可变更更新间隔被设置成0，即使默认更新间隔被设置成非0的数字，则该监控项目也不会采集监控数据，直至可变更更新间隔规则中所指定的时间区间结束，系统才会使用默认的更新时间间隔进行监控数据的采集。注意，如果被监控项目被配置成被监控设备代理主动模式时，则这个表单项中所填写的将被忽略。关于如何定义时间区间，请参考后续相关章节中的内容</p>

续表

表单项	描述
保留历史数据（Keep history, 单位天）	这个表单项用于配置Zabbix系统数据库中保留几天对应监控项目的监控数据，超过指定天数的对应监控项目的监控数据将自动被系统清除掉。很显然，这个表单项中所配置的数据越大，Zabbix系统数据库中历史表中的数据量就越大，相应的数据库所占用的磁盘空间就越大。因此，这个表单项里的内容应尽可能地设置得小一些，以减小Zabbix系统数据库中的数据量。而如果想了解某个监控项目较长的历史时间段内监控数据的变化情况，则可以将下面将要介绍的“保留趋势数据”表单项中的值设置得大一些
保留趋势数据（Keep trends,单位天）	对于数值型的监控项目，Zabbix系统不但保存该监控项目的历史数据，而且还保存该监控项目的趋势数据。至于历史数据和趋势数据有何联系和区别，将在后续章节进一步介绍。“保留趋势数据”表单项就是用于配置在Zabbix系统数据库中保存多少天的趋势数据的，超过期限的趋势数据系统都将会自动将其清除掉
取值方式（Store value）	这个表单项有3个可选项，分别为： 原始数据（As is）——表示对所采集的监控数据不作任何处理。 增量[变化/秒（Delta）]——表示系统取原始数据平均每ns的变化量作为监控数据，即当前监控数据=（当前系统所采集的原始监控数据 - 前一次系统所采集的原始监控数据）/（当前时间 - 前一次采集数据时的时间）。需要注意的是，如果系统当前所采集的原始数据小于前一次所采集的原始数据，则Zabbix系统将丢弃当前采集的原始数据，而等到下一次原始数据采集到后，再对数据进行计算。 增量[变化（simple change）]——即表示将系统当前所采集到的原始数据与前一次所采集到的原始数据之间的差值作为最终的监控数据
显示值（Show value）	这个表单项用于配置是否使用数据映射，以及配置当前的监控项目使用哪条数据映射规则。关于数据映射的规则、使用方法和配置，将在后续章节中详细介绍。值得注意的是，“显示值”表单项只能改变监控数据的显示方式，并不能改变对应数据在Zabbix系统数据库中的存储方式（数据库中存储的仍然是原始数据）。该表单项只对“信息类型”为整型的监控项目有效，对其他类型的监控项目无效
日志时间格式（Log time format）	我们知道，一般来说，软件和系统的日志信息都是带时间戳的。所以，当用Zabbix系统来监控系统或应用的日志信息时，“日志时间格式”表单项就用于配置当前监控项目接收到的日志信息中的时间戳的格式，Zabbix系统，使用该表单项配置的格式来匹配接收到的日志信息，并从中分离出日志信息中包括的时间戳信息。该配置项所支持的时间占位符有y：表示年（0001~9999）；M：表示月（01~12）；d：表示日（01~31）；h：表示小时（00~23）；m：表示分钟（00~59）；s：表示秒（00~59）；例如，如果将这个表单项里的内容填写为yyyyMMdd:hhmmss，则当系统接收到如“20140224:135850 Zabbix agent started.”形式的日志信息时，Zabbix系统就会将这条日志信息中的时间戳“20140224:135850”分离出来，并在显示这条监控信息时，将这条被分离出来的时间戳信息单独使用一个字段来显示。注意，上面“yyyyMMdd:hhmmss”配置内容中的“:”号只是占位符，它可以是除了“yMdhms”之外的任何可显示字符
新建监控分类（New application）	这个表单项的含义很简单，如果在这个表单项里输入了任何内容，则系统在创建监控项目的同时会自动创建一个新的监控分类，并将当前所创建的监控项目归为这个新建的监控分类

续表

表单项	描述
监控分类 (Applications)	该表单项用于选择，将新添加的监控项目归为哪个已存在的监控分类中
填充主机资产字段 (Populates host inventory field)	如果当前将要添加的监控项目所对应的监控数据，可作为主机资产中某个字段的内容，则可从这个表单项中选择该监控项目所关联的主机资产的字段。只有启用主机的自动填充主机资产功能后，这个表单项所设置的关联才起作用
状态 (Status)	这个表单项用于配置监控项目的状态。可选项有：“启用”，表示正常采集对应监控项目的监控数据；“禁用”，表示禁止对对应监控项目采集监控数据；“不支持”，表示对应监控项目不被Zabbix系统支持，这个状态一般由Zabbix系统自动设置，且Zabbix系统会根据其自身的配置，周期性地尝试将处于“不支持”状态下的监控项目的状态设置成“启用”状态，直至对应监控项目的状态不再被转变成“不支持”状态

注意：在添加监控项目时，“单位”表单栏里不可以输入 ms（毫秒）、转数每分钟（大写 RPM 和小写 rpm）、百分号（%）等符号。如果在该栏里填写了上面这些单位符号，系统虽然不会立即报错，但是，在显示相应数据时系统将会忽略这些单位符号。

3.6 配置触发器

在前面小节中，介绍了如何在一台主机上创建新的监控项目。从而我们知道，监控项目是监控数据的载体。然而，作为一个监控系统，仅仅只能采集监控数据是远远不够的，它至少还需要做到，当某个监控项目上所采集到的监控数据超过（或低于）设定的阈值时，能够自动通过某种方式，比如电子邮件、手机短信或即时通信工具等向相关人员发送报警信息，以便相关负责人员能够及时了解其所负责的主机出现的问题，从而能够及时地解决问题。所以，发现被监控主机出现故障，并及时通过某种方式向有关人员发送报警信息，是一款监控软件所必备的功能。而在 Zabbix 系统中，报警信息的发送动作是要靠触发器的触发来推动的。所以，在 Zabbix 系统中，为了使某台主机上的某个被监控项目在出现故障时，能够及时地发送报警信息，针对该项目创建和配置触发器是一个必须要完成的工作（在某些时候，我们可能并不需要对系统中的每个监控项目都创建触发器。因为有些监控项目，我们只是使用它来采集性能数据的）。下面，就来介绍一下如何为我们刚刚创建的监控项目创建一个触发器。实际上，或许我们称为某个监控项目创建触发器的提法并不是十分准确。因为，触发器和监控项目之间的关系并不是谁隶属于谁的关系。我们不能说某个触发器是隶属于某个监控项目的；同时，也不能说某个监控项目是隶属于哪个触发器的。但是，一个触发器往往是需要引用一个或多个监控项目（或者说引用监控项目上所采集的监控数据可能要更准确一点儿）的。同时，一个被监控项目也可以被多个触发器引用。

在具体介绍如何创建触发器之前，让我们来回顾一下，Zabbix 系统中的触发器的概念。

这里的触发器是基于监控项目所采集监控数据的逻辑表达式的，它定义了阈值，以判断监控项目上所接收到的监控数据是否出现了“问题”（并不是指接收数据的过程出现了问题，而

是指所接收到的监控数据超过或低于某个已定义的阈值，是一个“问题”数据）。在 Zabbix 系统中，当某个监控项目上所采集的监控数据，使得某个触发器的表达式为“真”时，则该触发器会被“触发”。只有触发器被触发了，才有可能将有“问题”的监控项目信息以报警的方式发送出去。

触发器只能处于两种状态中的一种状态下，这两种状态即为 OK 状态和 PROBLEM 状态。OK 状态即为正常状态；而一旦触发器处在 PROBLEM 状态，则表示该触发器被触发了。触发器的状态，会在每次 Zabbix 系统针对触发器表达式中所引用的监控项目，采集到新的监控数据时被重新计算。如果在触发器表达式中使用了 `nodata()`、`date()`、`dayofmonth()`、`dayofweek()`、`time()`、`now()` 等时间函数，则对应触发器的状态会每隔 30s 被 Zabbix 系统中的定时器重新计算一次。

创建触发器的步骤很简单，依次选择“系统配置”→“主机”菜单项。在主机列表的页面上，找到需要添加触发器的主机所在的行（在这里，选择我们在前面刚刚创建的主机），然后单击“触发器”列中对应的超链接，系统会打开触发器列表页面。在触发器列表页面上，单击页面右边上部的“新建触发器”按钮，系统将打开新建触发器的表单页面，如图 3-55 所示。

相对于创建新主机和创建新监控项目来说，创建新触发器页面上的表单项要少得多。创建新触发器页面上各表项的含义和作用如表 3-7 所示。

The screenshot shows the 'Create Trigger' form in Zabbix. It has a tabbed interface with 'Trigger' and 'Dependencies' tabs. The 'Trigger' tab is active. The form contains the following elements:

- 名称 (Name):** A text input field.
- 表达式 (Expression):** A large text area for the trigger expression, with an '添加 (Add)' button to its right.
- 表达式构造器 (Expression Builder):** A link below the expression field.
- Multiple PROBLEM events generation:** A checkbox.
- 描述 (Description):** A text area for the trigger description.
- URL:** A text input field.
- 警报级别 (Severity):** A dropdown menu with options: 未分类 (Unclassified), 一般信息 (Information), 警告信息 (Warning), 重要问题 (Important), 严重问题 (Critical), 灾难问题 (Disaster).
- 启用 (Use):** A checkbox.

图 3-55 创建触发器表单页面截图

在创建触发器页面的“名称”表单项里，输入要创建的触发器的名称，例如“3 分钟负载过高”等。然后，单击“表达式”文本框后面的“添加”按钮，系统会打开“触发条件”表单页面。单击这个页面“项目”表单项后面的“选择”按钮，在弹出的新页面上，找到上一节所创建的监控项目——CPU 负载，并选择它。接下来，在“函数”下拉菜单里找到并选择“Average value for period of T time > N”菜单项，同时，在“Last of (T)”表单项里输入 180，并在“N”表单项里输入“1”，最后单击“插入”按钮。完成上述操作后，系统将会返回到“配置触发器”页面，在这个页面的“表达式”文本框中添加上如“`{192.168.5.139:system.cpu.load.avg(180)}>1`”形式的触发器表达式。实际上，随着对 Zabbix 系统学习的深入，以及使用得越来越熟练，将来也可以不必每次在添加触发器表达式时都按照这里所述的方法，一步一步地选择项目、函数等，而是可以直接在“表达式”文本框里输入需要书写的表达式。最后，单击页面下方的“保存”按钮，以保存为触发器配置的信息，这样就在系统中成功地创建了一个新的触发器。

提示：关于编写触发器表达式应遵循的规则和其他相关要求，将在下一章中详细介绍。

表 3-7 配置触发器表单项列表

表单项	描述
名称 (Name)	该表单项用于填写触发器的名称，该名称即为在其他页面，例如触发器列表页面等，显示触发器时系统使用的触发器名称。与监控项目的名称类似，触发器名称中也可以支持\$1、\$2、\$3、……、\$9等形式的宏变量。只是，触发器名称的宏变量\$1、\$2、\$3……所替代的内容是触发器表达式中第1个、第2个、第3个……第9个常量。例如，如果将上面所配置的触发器名称修改为“3分钟负载高于\$2”，则在显示触发器列表时，对应触发器的名称就会变成“3分钟负载高于1”。因为上述触发器表达式中右边的常量是1，所以，系统在显示该触发器名称时，将会用这个常量“1”来替换掉名称中的宏变量\$1
表达式 (Expression)	用于计算触发器状态的逻辑表达式。关于触发器表达式的书写规则和要求，我们将在后续章节中详细介绍和说明
生成多个问题事件 (Multiple PROBLEM events generation)	这个表单项是一个复选框。如果在创建触发器时，不选中这个复选框，则表示只在触发器状态发生改变时，系统才会产生一个事件。即，当触发器由OK状态转变为PROBLEM状态或者触发器的状态由PROBLEM 状态转变为OK状态时，系统都会产生相应的事件。而当触发器持续处在某个状态时，例如PROBLEM状态，则只要系统所采集到的新的监控数据使触发器维持在之前的状态，系统便不会再生成与该触发器相关的事件，只有当这个触发器的状态改变时，比如变为OK状态时，系统才会生成新的事件。而如果选中这个复选框，则情况完全不同。不仅触发器的状态改变时，系统会生成一个与该触发器相关的事件，而且一旦某个触发器处于PROBLEM状态下，则系统后续每采集到一次监控数据，只要这个数据仍然使对应触发器维持在PROBLEM状态下，系统都会生成一个新的与该触发器相关的事件。而当触发器维持在OK状态时，则只要系统所采集到的监控数据不使相关触发器的状态改变成PROBLEM状态，则系统就不会生成与对应触发器相关的新事件。而在Zabbix系统中，事件是用来驱动系统发送报警信息的信息源。因此，如果在创建触发器时，选中这个复选框，则该触发器一旦处在PROBLEM状态下，系统每新采集一次对应触发器引用的数据，并且使对应触发器维持在PROBLEM状态下，系统就会发送一次报警信息
描述 (Comments)	这个表单项里可以输入关于触发器的描述或说明性的文本内容，例如，触发器被触发时的处理方法和操作步骤、负责对应触发器管理和维护人员的联系方式，等等
URL	如果在创建触发器时，在这个表单项里输入了内容，则当通过“状态统计”→“触发器”菜单项所对应的页面显示触发器状态时，单击触发器名称上的超链接，系统就会在弹出的菜单上显示出以我们在这个表单项里所输入的内容为链接的URL菜单项。单击这个URL菜单项，就可以打开与URL对应的页面。URL表单项的内容支持{TRIGGER.ID}宏，但也仅只支持这一个宏，其他宏均不被支持

续表

表单项	描述
警报级别 (Severity)	在这里选择我们新创建的触发器的级别。我们在前面章节中已经介绍过，Zabbix 系统中触发器可以分为 6 个级别。这 6 个级别分别是：“未分类 (Not Classified)”、“一般问题 (Information)”、“警告问题 (Warning)”、“重要问题 (Average)”、“严重问题 (High)”和“灾难问题 (Disaster)”。这个表单项就是用于指定，我们将要创建的触发器所属于的级别的。实际上，触发器的这 6 个级别的名称，我们自己是定义的，后续章节将介绍如何定义触发器的名称及其相应的显示颜色

3.7 接收第一条报警信息

前面我们已经创建了一台被监控主机，并在这台主机上创建了一个监控项目和一个触发器。接下来，需要配置发送报警信息的“消息介质”，以及执行报警信息发送的“动作”及相关的规则。完成这些配置之后，就可以测试我们所做的配置是否正确，并让系统向我们指定的邮箱里发送第一封报警邮件。通过对上述这些对象的配置和实际测试，可以使我们对 Zabbix 系统中事件的产生、报警信息的发送等过程有一个大概的理解和掌握。

3.7.1 配置 Email 消息介质

前面提到过，Zabbix 系统安装部署完成之后，系统会预设“Email”、“SMS”、“Jabber”等 3 种消息介质。在这 3 种消息介质中，“SMS”是用来发送手机短信的，但是需要在 Zabbix 服务器上接上 GSM 猫，且需要安装配置好相应的驱动程序。现今，一般很少使用这种方式来发送报警短信。这是因为，Zabbix 服务器一般存放在 IDC 机房的机柜里，而将移动通信设备放置在 IDC 机房里的机柜里，移动信号通常都比较差，因此就必然会影响报警短信发送的可靠性。而 Jabber 是用来发送 IM（即时信息）信息的。如果使用 Jabber 发送报警信息，则需要有相应的 IM 软件客户端，我们才能接收到系统发送过来的报警信息。而且据了解，在国内支持 Jabber 的客户端软件并不多。所以，在国内通过 Jabber 方式发送报警信息的单位并不多。最后一种系统预设消息介质就是 Email，电子邮件是我们现在日常办公中最常见的一种通信工具了，所以我们当然需要让 Zabbix 系统能够支持通过 Email 方式发送报警信息。但是，Zabbix 系统中预设的“Email”消息介质是不支持 SMTP 协议认证的。而在垃圾邮件泛滥成灾的今天，估计很少有电子邮件服务器还开放着邮件中继，而不需要 SMTP 验证的吧。所以，需要创建一种新的 Email 消息介质，并且使它能够支持我们电子邮件服务器所要求的，在发送电子邮件时能够进行用户验证。

1. 安装配置 msmtplib 软件包

msmtplib 软件是 SMTP 服务的客户端工具，它可以按用户要求向邮件服务器发送电子邮件。这里我们将使用 msmtplib 和 mutt 软件包发送邮件报警信息。以下是 msmtplib 软件的安装配置过程：

```
shell>
wgethttp://sourceforge.net/projects/msmtplib/files/msmtplib/1.4.30/msmtplib-1.4.30.tar.bz2/
```

```

download                                #下载 msmtplib 软件包
shell> tar -jxvf msmtplib-1.4.30.tar.bz2  #解压所下载的软件包
shell> cd msmtplib-1.4.30
shell> ./configure --prefix=/usr/local/msmtplib  #执行编译配置
shell> make && make install  #编译并安装 msmtplib 软件
shell> mkdir -p /usr/local/msmtplib/etc  #创建配置文件目录

```

#创建配置文件 msmtplibrc, 其内容如下

```

shell> vi /usr/local/msmtplib/etc/msmtplibrc
1. account default  # 配置默认账号
2. host smtp.domain.com  # SMTP 服务器地址或主机名
3. port 25  # SMTP 服务的端口, 默认为 25
4. from zabbix@domain.com  #发送邮件的电子信箱地址
5. auth login  # SMTP 验证方式, 一般选 login
6. tls off  # 关闭 SSL
7. user zabbix@domain.com  #发送邮件的用户名
8. password zabbix  #用户密码
#msmtplib 软件的日志文件路径及文件名
9. logfile /var/log/zabbix/msmtplib.log

```

接下来, 用 yum install mutt 命令安装 mutt 组件。安装完成后, 修改 mutt 的配置文件/etc/Mutttrc 中的内容, 修改后的/etc/Mutttrc 文件内容如下:

```

1. set sendmail="/usr/local/msmtplib/bin/msmtplib"  #指定 msmtplib 命令工具所在的绝对路径
2. set use_from=yes
3. set realname=zabbix@domain.com
4. set editor="vim"
5. set charset="utf-8"
6. set rfc2047_parameters=yes

```

完成上述配置后, 可以使用 echo "test mail" | mutt -s "test" yourname@domain.com 命令来测试我们做的安装配置是否正确。如果执行上述命令后, 邮箱 yourname@domain.com 中能收到一封主题为“test”, 内容为“test mail”的邮件, 则说明我们做的安装配置正确。否则, 需要认真检查是哪里安装配置不正确。

2. 编写邮件发送脚本程序

如果上述的测试均正确, 则接下来需要编写一个简单的 shell 脚本, 以供 Zabbix 系统调用来发送报警邮件。编写发送邮件脚本的命令及脚本程序内容如下所示:

```

shell> vi /opt/zabbix/share/zabbix/alertscripts/sendemail.sh
1. #!/bin/bash
2. echo "$3" | mutt -s "$2" $1 # $3

```

这个脚本中的参数\$1、\$2、\$3 均是 Zabbix 系统在调用这个脚本时, 由 Zabbix 系统传递给它的。其中, 参数\$1 为邮件接收者的邮箱地址; 参数\$2 为邮件的主题内容; \$3 则为邮件的正文内容。

将脚本文件设置成可执行文件

```
shell> chmod +x /opt/zabbix/share/zabbix/alertscripts/sendemail.sh
```

#将这个脚本的属主和属组分别设置成 zabbix 用户和 zabbix 用户组

```
shell> chown zabbix:zabbix /opt/zabbix/share/zabbix/alertscripts/sendemail.sh
```

3. 配置 Email 消息介质

在配置 Email 消息介质之前, 首先需要确认 Zabbix 服务器端组件配置文件/opt/zabbix/etc/zabbix_server.conf 中的 AlertScriptsPath 配置项所配置的路径是正确的。如果这个配置项被注释

掉了，或者所配置的路径与存放发送邮件的脚本文件 `sendemail.sh` 的路径不一致，则需要将这个配置项前的注释符去掉，并将其所配置的路径设定为 `sendemail.sh` 脚本存放的路径，并重启 `Zabbix_server` 组件进程。

要配置 Email 消息介质，则需在 Web 页面上依次选择“高级配置”→“消息介质”菜单项（需要有超级管理员权限）。在“配置消息介质”页面上，单击页面右上边的“新建消息介质”按钮，系统就会打开创建新消息介质的页面，如图 3-56 所示。



图 3-56 创建新消息介质页面截图

创建新消息介质表单的内容比较简单，只需在“描述”表单项里输入我们给新创建的消息介质所取的名字，这里输入 `Sendemail`；“类型”表单项选择“脚本”；在“脚本名称”表单中输入我们刚刚编写的脚本文件的文件名——`sendemail.sh`。需要注意的是，这里并不需要输入该脚本所存放的路径，而只需输入脚本的文件名即可；“启用”复选框当然是需要勾选的，然后单击“保存”按钮，以保存我们所创建的消息介质的配置信息，这样就创建了一个新的消息介质。

接下来，需要将这个新添加的消息介质应用给 Zabbix 系统用户。方法是，依次选择 Web 页面上的“高级配置”→“用户”菜单项。然后，在“配置用户及用户组”页面上的“成员”列里，找到需要应用新消息介质的用户。这里，建议至少要将这个新创建的消息介质应用给 `Admin` 用户。然后单击找到的用户账户名上的超链接，系统将打开“配置用户”页面。接下来的操作方法就与我们在前面 3.1.2 节中介绍的操作是一样的了，请用户自行参考 3.1.2 节的介绍，将新添加的消息介绍应用给指定的用户。

3.7.2 配置手机短信消息介质

上面配置了一个用于发电子邮件的消息介质。但是，在很多时候，我们希望所监控的系统只要一出现严重的问题，就要及时得到报警信息。很显然，通过电子邮件发送报警信息的方法，很难满足我们对报警信息的及时性需求。这个时候，如果能让 Zabbix 系统给指定的用户手机上发送手机短信报警是一个不错的选择。虽然，当安装部署好 Zabbix 系统后，系统会预配好了 SMS 消息介质，且该消息介质也是可以用来发送手机短信的，只是该消息介质调用的是，直接连接到 Zabbix 服务器上的 GSM 猫接口。所以，这个系统默认配置好的消息介质，现在一般很少使用。因为我们的监控服务器一般是安放在 IDC 机房里的，而依据我们实际的工作经验，一般 IDC 机房里的无线信号很不稳定。这样，如果直接使用这个消息介质来发送报警短信的话，那么可靠性就很难得到保证。

如今，市场上有很多能为用户提供手机短信发送服务的服务提供商。如果用户向这些服务

提供商购买手机短信发送服务，它们一般会向用户开放一个适用于程序调用的手机短信发送网关接口。通过调用这个接口，就可以很容易地实现向指定用户手机上发送手机短信的功能。不同的手机短信发送服务商，所提供的短信网关接口，其调用方法和接口参数可能不尽相同。但是，如今，很多服务商提供的短信网关接口都支持 Web service 方法调用。所以，它们的调用方法也都大同小异。一般只需依照服务商提供给我们的技术资料上的说明，编写相应的程序，按规范调用指定接口，就可以实现手机短信的发送。下面，我们提供一个调用短信网关接口的例程，供读者参考。

```

1. #!/bin/env python
2. #coding:utf-8
3. ###sendsms.py
4. ###请将本程序保存为<zabbix_install_root>/share/zabbix/alertscripts/sendsms.py 文件
5. ###
6. try:
7.     import urllib2
8.     import sys
9.     import string
10. except:
11.     print -1
12.     sys.exit(-1)
13. if len(sys.argv)<3:
14.     print -1
15.     sys.exit(-1)
16. msg=""
17. base = [str(x) for x in range(10)] + [ chr(x) for x in range(ord('A'),
ord('A')+6)]
18. url=http://esms.etonenet.com/sms/mt?command=MT_REQUEST&spid=12345&sppass
word=67893456&da=86
19. def dec2hex(string_num):
20.     num = int(string_num)
21.     mid = []
22.     while True:
23.         if num == 0: break
24.         num,rem = divmod(num, 16)
25.         mid.append(base[rem])
26.     return ''.join([str(x) for x in mid[::-1]])
27. for char in sys.argv[2]:
28.     msg=msg+dec2hex(ord(char))
29. msg=msg.upper()
30. url=url+sys.argv[1]
31. url=url+"&sm="+msg+"&dc=15"
32. print url
33. fp=urllib2.urlopen(url)
34. line=fp.readlines()

```

与配置电子邮件消息介质一样，请将上述脚本程序保存为/opt/zabbix/share/zabbix/alertscripts/sendsms.py 文件，并将该脚本文件设置成可执行文件，且需将该文件的属主和属组设置成 zabbix。完成上述设置之后，就可以通过 Web 前端组件来配置手机短信消息介质了，配置方法与配置 Email 消息介质类似。

3.7.3 创建新动作

前面已经提到过，在 Zabbix 系统中，当一个触发器被触发后，会推动动作执行，从而发送报警信息。因此，要想让 Zabbix 系统能够在被监控主机或项目出现问题的时候给我们发送报警信息，就需要在系统中创建动作。创建动作的方法比较简单，在 Web 前端页面依次选择“系统配置”→“动作”菜单项。在配置动作页面上，单击页面右上部的“新建动作”按钮，系统将打开配置动作表单页面，如图 3-57 所示。

动作

触发条件

行为

名称

admin

默认升级周期(最小60秒)

3600 (秒)

默认主题

{TRIGGER.STATUS}: {TRIGGER.NAME}

默认消息

Trigger: {TRIGGER.NAME}
Trigger status: {TRIGGER.STATUS}
Trigger severity: {TRIGGER.SEVERITY}
Trigger URL: {TRIGGER.URL}
Item values:

恢复消息

☐

启用

☒

保存

取消

图 3-57 配置动作表单截图

从图 3-57 中，可以看出，在配置动作表单页面上有“动作(Action)”、“触发条件(Conditions)”和“行为(Operations)”3 个选项卡。通过“动作”选项卡，可以配置动作的名称、默认主题和默认消息等内容。该选项卡上各表单项的含义和作用如表 3-8 所示。而通过“触发条件”选项卡，可以创建和配置动作被执行的条件和规则。需要说明的是，这里的“触发条件”并不是指触发器被触发的条件，而是指动作被执行的条件。实际上，触发器是否会被触发的条件是我们创建触发器时，通过触发器的表达式指定的。在 Zabbix 系统中，动作的触发条件可以被配置成比较复杂的规则，关于这方面的内容，将在后续章节中再作详细的介绍。配置动作表单页面上的第 3 个选项卡是“行为”选项卡，其用来配置动作具体执行什么行为，比如，是给指定的用户或用户组的所有人员发送报警信息，还是执行某条指定的命令。“行为”选项卡上的表单项内容如图 3-58 所示。

表 3-8 中列出了“动作”选项卡上各表单项的含义和作用。然而，这里仅是为了介绍创建一个新动作的一般过程，所以，无须填写这个选项卡上的每一个表单项，而仅需要在“名称”表单项里输入新建动作的名称，例如 admin，其他表单项的内容都保持系统默认值即可。完成动作名称的输入之后，切换到“行为”选项卡上，以完成动作的行为配置，如图 3-58 所示。

表 3-8 “动作”选项卡表单项列表

表单项	描述
名称 (Name)	系统中唯一的动作名称。当在系统中查看动作列表时，系统所显示的动作名称就是通过这个表单项配置的

续表

表单项	描述
默认升级周期 (最小60秒)	在Zabbix系统中，一个动作的行为是可以升级的。这是一个什么概念呢？其实就是动作可以被配置成，在不同的时间段内将报警信息发送给不同级别的人员，或者执行不同级别的远程命令。“默认升级周期”表单项就是用来配置动作升级的时间间隔的。之所以叫做“默认升级周期”，是因为在“行为”选项卡表单上，也可以针对不同的行为级别，指定不同的升级时间间隔。如果在“行为”选项卡里没有针对不同的行为级别，指定不同的升级周期，则系统将会以默认升级周期作为动作行为升级的周期。如果针对动作行为没有指定不同的行为级别，则系统将以默认升级周期为间隔重复发送报警信息或者重复执行远程命令，直至故障被消除。当然，可以指定重复发送报警或重复执行远程命令的次数。从这一点上，我们或多或少可以看出Zabbix系统的功能是非常强大的
默认主题 (Default subject)	默认的信息主题。在前面，我们在介绍配置Email消息介质时，提到过，在Zabbix系统调用Email消息介质对应的脚本时，会给这个脚本传递3个参数，分别为\$1、\$2和\$3。“默认主题”表单项里所配置的内容，将会被Zabbix系统以参数\$2的形式传递给发送邮件的脚本程序。“默认主题”里的内容可以支持宏变量，例如图3-57中的{TRIGGER.STATUS}和{TRIGGER.NAME}，就是触发器状态和触发器名称所对应的宏变量。当动作被触发时，系统将会以实际被触发的触发器名称和触发器状态替换掉这两个宏变量，并将替换后的内容作为最终的主题传递给相应的脚本程序。“默认主题”表单项中可以支持的系统宏变量列表请参见本书附录D。之所以叫做“默认主题”，是因为在配置动作的行为时，可以针对不同的用户或用户组配置不同的消息主题。关于如何在配置行为时，针对用户或用户组的不同而使用不同的消息主题，将在后续章节中详细介绍
默认消息 (Default message)	默认的信息。与默认主题类似，当动作被触发时，这里所配置的信息内容将会被系统作为参数传递给指定的脚本程序。同样，“默认消息”表单项里也可以添加系统所支持的宏变量
恢复消息 (Recovery message)	在前面，我们提到过，当一个触发器的状态从PROBLEM状态转变为OK状态，或者从OK状态转变为PROBLEM状态时，系统都会产生相应的事件。当在“动作”选项卡中勾选上“恢复消息”时，则系统会将触发器状态从PROBLEM状态转变为OK状态时所产生的事件视为恢复事件，并给指定的用户或用户组的用户发送一条恢复消息。需要注意的是，恢复消息只会发送一次，不会像PROBLEM消息那样会执行升级或周期性的重复发送。同时，如果要想让用户收到恢复消息，则要在动作条件中将“触发状态=‘OK’”这条添加到触发条件中。否则，即使系统产生了恢复事件，该事件也不会触发动作，则用户当然也就没有办法收到恢复消息了。最后，恢复消息只会发给那些接收到了PROBLEM状态信息的用户。换句话说，如果某个触发器的触发引起动作的触发并没有升级到更高级别，那么当它恢复时，恢复消息也不会升级
恢复主题 (Recovery subject)	当勾选上“恢复消息”复选框后，系统将会在“动作”选项卡的表单中增加两项表单项：恢复主题(Recovery subject)和恢复消息(Recovery message)。“恢复主题”表单项的含义与“默认主题”表单项的含义是类似的，即在发送动作恢复信息时，将这里所填写的内容作为恢复的信息主题发送给用户。同样，“恢复主题”表单项也是支持系统宏变量的

续表

表单项	描述
恢复消息 (Recovery message)	其用法和含义与“默认消息”的用法和含义是类似的，且它也可以支持宏变量
启用 (Enabled)	当勾选上这个复选框时，表示启用对应动作，否则表示禁用对应动作

动作行为

步骤

详细

周期(秒)

延迟

动作

1

发送消息到用户: Admin

默认

立即

编辑

移除选中项

行为详细

间隔

起始

1

截止

1 (0 - infinitely)

升级周期

0 (最小60秒, 0 - 使用默认值)

行为类型

发送消息

发送消息到用户组

添加

发送消息到用户

添加

发送消息方式

- 全部 -

默认消息

☒

触发条件

没有定义条件

新建

添加

取消

图 3-58 行为配置选项卡截图

相对于“动作”选项卡上的表单内容，“行为”选项卡上的表单内容要复杂一些，这个选项卡上各个表单项的含义和作用如表 3-9 所示。

表 3-9 详细列出了“行为”选项卡中每个表单项的含义和作用。假如你只是简单地阅读一下对这些表单项的作用和含义的描述，可能并不能真正对它们的含义和作用有很深的理解和体会。所以，建议你在学习过程中，对这些表单项亲自动手多做一些测试，才会对这里的介绍有较深的理解和掌握。

表 3-9 “行为”选项卡表单项列表

表单项	描述
动作行为 (Action operations)	<p>“动作行为”数据区域列出的是，当前动作中已经创建的行为列表。它包括以下内容：</p> <p>步骤 (Steps) —— 行为所对应的步骤区间范围。即在这个字段中所指定的步骤范围内，动作执行其后面“详细”字段中所指定的行为</p> <p>详细 (Details) —— 显示行为的类型，即指发送信息给用户还是执行远程命令</p> <p>周期 (Duration) —— 显示对应步骤的升级周期。如果对应步骤没有配置升级周期，则这里显示的是默认升级周期，否则这里显示的是对应步骤所配置的升级周期</p> <p>延迟 (Start in) —— 显示对应步骤的行为，在事件发生后延迟多久被执行</p> <p>动作 (Action) —— 单击该字段对应的超链接，可以编辑对应的行为</p> <p>该数据区域内还有“新建”和“移除选中项”两个超链接，分别单击这两个超链接，可以创建新的行为或删除一个存在的行为</p>

续表

表单项	描述
行为详细 (Operation details)	<p>单击“动作行为”数据区域中的“新建”超链接，或单击某个行为列表后面的“编辑”超链接，系统将展开“行为详细”数据区块。在“行为详细”数据区块内，可以配置行为的详细信息。当在“行为详细”数据区块的“行为”表单项里分别选择“发送消息”和“远程命令”时，则“行为详细”数据区块里所显示的表单项并不完全相同。在本表以下部分中，我们分别介绍了这两种行为类型所对应的表单项内容及其含义，但是不具体地标注某个表单项是在选择“发送消息”选项时系统所显示的表单项，还是在选择“远程命令”选项时所显示的表单项。读者在阅读本书时，结合实际系统一看便知这些表单项分别显示在哪种类型行为表单上</p>
间隔 (Step)	<p>用于指定当前所配置的行为对应于哪个升级步骤区间：</p> <p>起始 (From) —— 指定当前所配置的行为从哪个升级步骤开始执行。</p> <p>截止 (To) —— 指定当前所配置的行为到哪个升级步骤停止。当这个表单项被填写为0时，则表示当前所配置的行为一直被执行下去，直至所对应的故障恢复为止。换句话说，如果在“截止”表单项里填写数字0，则表示在每个升级周期，系统都会向指定的用户或用户组里包含的全部用户发送一次报警信息，或者执行一次指定命令，直到所对应的故障解除。</p> <p>升级周期 (Step duration) —— 与默认升级周期是相对的。这里所定义的升级周期是针对所定义的升级步骤区间里的升级步骤而言的。如果这个表单项里输入0，则表示使用默认的升级周期。当多个行为有相同的升级步骤区间 (Zabbix系统是允许这么做的) 时，则系统将这些行为中，最小的升级周期作为对应升级区间内各步骤的升级周期</p>
行为类型 (Operation type)	<p>该表单项是一个下拉菜单，用于选择当前所配置的行为属于哪种类型。可选的类型有：</p> <p>发送消息 (Send message) —— 如果选择这个选项，则对应动作在当前所指定的升级步骤区间内的行为是，发送报警信息给指定的用户或指定用户组内的每一个用户。</p> <p>远程命令 (Remote command) —— 如果选择这个选项，则对应动作在当前所指定的升级步骤区间内，执行指定远程命令</p>
发送消息到用户组 (Send to user groups)	<p>用于配置哪些用户组里的用户，可以接收由当前行为所发出的报警信息。单击“添加”超链接，可以添加消息接收的目标用户组；单击“移除”超链接可以删除指定的目标用户组。需要注意的是，要使目标用户组里的每一个用户都能收到当前行为所发出的报警信息，则对应用户组里的每一个用户至少都需要具有对应主机的读取权限，否则系统将不会将相应报警信息发送到对应用户</p>
发送消息到用户 (Send to users)	<p>与“发送消息到用户组”表单项的功能和含义是类似的，只是这个表单项是用于指定接收消息的目标用户的而不是用户组。类似的，指定的用户也需要具有对发生故障的主机的可读权限，否则即使我们在这里配置了将报警信息发送给他，系统仍然无法将对应的报警消息发送给该用户</p>
发送消息方式 (Send only to)	<p>指定将报警信息发送到系统中设置的所有消息介质上，还是选定的一个或多个消息介质上</p>
默认消息 (Default message)	<p>如果勾选上这个复选框，则系统将使用我们之前在“动作”选项卡上所配置默认主题和默认消息作为报警信息的主题和内容发送给目标用户。如果取消这个复选框的勾选状态，则系统会显示“主题”和“消息通知”这两个表单项</p>

续表

表单项	描述
主题 (Subject)	当取消“默认消息”复选框的勾选状态时，系统就会使用这个表单项。其作用和用法与在“动作”选项卡中配置的“默认主题”表单项的作用和用法是类似的。只是这个表单项里所配置的内容只对当前所配置的行为有效
消息通知 (Message)	与主题类似，用于配置当前行为在执行时发送给用户的消息内容。与配置“默认消息”表单项一样，“消息通知”里的内容是可以使用系统宏变量的。“消息通知”里所配置的宏变量，将在系统调用具体消息介质时，使用具体内容进行替换
目标列表 (Target list)	当“行为类型”选择为远程命令时，系统就会显示出“目标列表”这个表单项。这个表单项用于配置远程命令执行的目标主机。当单击这个表单项数据区块里的“添加”超链接时，可以选择添加当前主机、其他主机或主机组，作为远程命令执行的目标主机。单击已经存在的目标主机列表后面的“移除”超链接，可以从列表中删除选定的主机或主机组
类型 (Type)	用于指定远程命令的类型，可以是IPMI命令、自己编写的脚本程序或通过SSH方式连接到远程主机上所执行的远程主机上的系统命令，以及telnet方式所执行的远程主机上的系统命令或者脚本程序。选择不同的命令类型，其所对应的与执行具体命令相关的配置也不同，所以系统呈现出的表单项也不一样。例如，当选择“SSH”选项时，系统就会显示出与SSH连接相关的参数配置表单项。这些表单项的内容和含义都比较简单，相信读者一看表单项的名称就知道它们的作用，故在这里就不对它们一一进行介绍了
命令 (Commands)	这个表单项里当然应该填写行为中被执行的具体命令
行为条件 (Conditions)	配置行为在什么条件下执行，即是在事件未被确认条件下还是在事件已经确认的条件下执行

单击行为数据块中的“新建”超链接，系统将展开“行为详细”数据区块的表单项。“起始”表单项里填写 1，“截止”表单项里填写 0。单击“发送消息到用户”后面的“添加”超链接，并选择接收报警信息的目标用户，其他内容均保持系统默认值。接下来，单击“行为详细”数据块内的“更新”超链接，以保存我们对行为所作的配置。这样，就完成了动作行为的配置。

完成上面所述的“动作”和“行为”选项卡上相应内容的配置后，单击页面上的“保存”按钮，以保存我们对动作的配置。这样，就在系统中创建了一个新动作。

3.7.4 接收第一条报警信息

前面我们在 Zabbix 系统中添加了一台被监控主机、一个监控项目并配置了一个触发器，安装和配置好了 Email 和手机短信消息介质，同时在系统中配置了一个用于发送报警信息的动作。接下来，就需要来验证一下我们之前所做的配置是否正确，即实际验证当被监控主机的平均负载连续 3 分钟超过 2 时，Zabbix 系统是否能如我们所预期的那样，给指定的邮箱里发送报警信息。为此，可以在被监控主机上执行多条如下命令，人为造成被监控主机的平均负载连续 3 分钟超过我们在触发器里配置的阈值 2。具体需要执行多少条这样的命令，才能使被监控主机的平均负载连续 3 分钟超过 2，要依据你的主机的硬件配置高低而定。总之，需要让被监控主机上的平均负载超过 2，并且这样的状态要维持超过 3 分钟以上，以便使所配置的触发器被触发，从而使我们能够接收到所预期的报警信息。


```
shell> cat /dev/urandom | md5sum &
```

通过我们在前面章节中所介绍的如何查看主机的最新监控数据, 如何查看触发器状态以及如何查看事件的方法, 可以实时查看 Zabbix 系统所采集到的监控项目的最新监控数据、触发器是否被触发、何时被触发以及触发器被触发而生成的事件信息等。

如果之前所做的所有配置都是正确的, 且 Zabbix 系统工作正常, 则几分钟之后, 我们在用户消息介质里配置的邮箱, 就会接收到一封主题类似于“cpu load 大于 2:7.59:南汇监控服务器 (1192.168.5.139):system.cpu.load”的报警邮件。同时, 当查看 Zabbix 服务器上邮件发送日志文件/var/log/zabbix/msmtp.log 里的内容时, 也会看到有邮件被发送出去的日志信息。

- 提示: 1. 如果没有收到预期的报警邮件, 则请确认该报警邮件是否在你的邮件垃圾箱里。某些邮件系统可能会将 Zabbix 系统所发出的邮件存放到邮件垃圾箱里。
2. 确认所添加的消息介质配置是否正确。
 3. 确认用户消息介质里所配置的邮件地址是否正确。
 4. 确认触发器配置是否正确, 以及是否被触发, 等等。
 5. 想中断“cat /dev/urandom &”命令的执行, 可以执行“killall cat”命令。

3.8 本章小结

在本章, 我们详细介绍了 Zabbix 系统 Web 组件中各种查看类菜单的功能和作用, 以及它们对应页面所显示内容的含义。并在 Zabbix 系统中添加了一个新用户, 配置了第一台被监控主机, 同时在新添加的被监控主机上添加了监控项目, 且针对这个监控项目配置了一个触发器。最后还安装配置了发送邮件的消息介质, 创建了用于发送报警信息的动作。在完成上述这些安装和配置之后, 人为地造成了被监控主机上的 CPU 负载超过我们设置的报警阈值, 从而使我们能够接收到从 Zabbix 系统发出来的第一条报警信息。

通过本章的介绍, 你大概对 Zabbix 系统的配置有了一个初步的了解和掌握。但是, 本章包含了大量配置项和页面表单项含义及作用的说明。对于这些配置项和页面表单项的含义和作用, 你可能一时无法完全理解和掌握, 这就需要多做一些实际测试和研究。只有通过实际测试, 才能更深刻地理解和掌握这些配置项和页面表单项的含义和作用。

第4章 Zabbix 系统中相关规则及原理

通过前一章的介绍，我们对 Zabbix 系统，特别是对 Zabbix 系统的 Web 前端组件有了一个初步的认识和掌握。但是，对于在使用和配置 Zabbix 系统时，我们应该遵守的规则以及 Zabbix 系统中一些功能的其本原理还不甚了解。例如，我们在前一章介绍如何创建监控项目时提到过，监控项目的关键字很重要，当在 Zabbix 系统中引用一个监控项目时，所引用的就是这个监控项目的关键字。而且，我们在前面章节中不止一次提到过，在 Zabbix 系统中，对不同类型监控项目的关键字有不同的配置要求。比如，某些类型监控项目的关键字可以自由输入，而另外一些类型的监控项目，其关键字的形式又是相对固定的，我们不可以随便指定。其实，在 Zabbix 系统中，不管是可以自由定义的还是具有相对固定形式的监控项目关键字，它们的命名都需要遵守一定的命名规范。

另外，通过前面章节的介绍我们知道，在 Zabbix 系统中的许多地方都可以使用宏变量。其实，在 Zabbix 系统中，不仅仅可以在许多地方使用 Zabbix 系统中预定义好的宏变量，而且还可以使用我们自己定义的宏变量，且对于这些自定义的宏变量，我们既可以将它们定义成系统全局的，也可以将它们定义在某个模板上，还可以将它们定义在某台主机上，而且在这 3 个层级上所定义的变量是可以同名的。既然在 Zabbix 系统中，可以分别在全局、模板和主机层面上定义名称相同的宏变量，那么系统在使用这些宏变量时，将以怎样的顺序来使用它们呢？

最后，在前面各章节中，虽然陆续对 Zabbix 系统中的事件、触发器、动作和消息介质等做过一些介绍和阐述。但是，在 Zabbix 系统中，这几种对象之间到底是什么关系呢？在 Zabbix 系统中，从事件的产生到报警信息的发出，这之间的流程是什么样子的呢？本章我们将就上述的这些问题展开较深入的讨论。

4.1 监控项目关键字命名规范

前一章，我们在介绍如何创建监控项目时提到过，Zabbix 系统中的监控项目关键字，可以从系统中选择预定义好了的；还可以由我们自己输入。但是，不管是从系统中选择预定义的监控项目关键字，还是我们自己输入关键字，在配置它们的时候都必须遵循一定的格式和规范。

4.1.1 监控项目关键字命名规范

在 Zabbix 系统中，监控项目关键字的组成格式为：关键字名称[参数 1,参数 2,参数 3……]。例如，`zabbix[wcache,history,total]`、`zabbix[wcache,values,not supported]`、`system.sw.arch`、`hrStorageUsedInBytes[/]`，等等，都是合法的监控项目关键字。在监控项目关键字中，关键字名称必须要有，且关键字名称只能由 0-9、a-z、A-Z、_、-、. 等字符组成。具体来说，关键字名称只能由数字、大小写英文字母、下划线、连接符以及半角的句号（.）组成。由此可知，监控项目关键字名称不支持中文以及除上述字符以外的其他特称字符，例如 @、#、(、) 等字符。但是，与其他许多高级编程语言所规定的变量的命名规则不一样的是，在 Zabbix 系统中，监控项目关

键字的名称中的第一个字母可以是数字。

另一方面, 监控项目关键字中可以包含关键字参数, 但是所有的参数都必须填写在[]号之中, 且多个参数之间需要用半角的逗号(,)隔开。例如上例中的“wcache”、“history”、“/”等, 就是关键字的参数。关键字参数可以用引号引起来的字符串, 也可以是未被引号引起来的字符串, 甚至可以是数组。且当关键字中包含多个参数时, 可以省略某些参数。但是, 如果省略的参数后面, 还有未被省略的参数, 则分隔不同参数之间的逗号不能省略。例如, 在前面章节介绍的“简单检查”数据采集方法中, 有一类用于检查主机是否存活的监控项目, 它们的监控项目关键字形式是 `icmpping[<target>,<packets>,<interval>,<size>,<timeout>]`。从这个关键字的形式中, 可以看出完整的关键字包含的参数有 5 个, 它们分别是 `target`: 用于指定被检查目标主机的 IP 地址或其合法的 DNS 主机名; `packets`: 用于指定每次检查时, Zabbix 服务器或代理服务器发送的 ICMP 数据包个数; `interval`: 用于指定相邻两次 ping 检查的执行时间间隔(单位是 μs); `size`: 用于指定每次发送的数据包的大小(单位是字节)以及 `timeout`: 用于指定检查的超时时间, 单位也是 μs 。对于该类监控项目某个具体监控项目的关键字, 如果将其写成 `icmpping[,200,,500]` 形式, 是完全合法的。通过上面的介绍, 我们知道当将某个监控项目的关键字写成 `icmpping[,200,,500]` 形式时, 则表示该监控项目的关键字中省略了 `target`、`packets` 和 `size` 参数。当某个监控项目关键字的参数被省略时, 则对于所省略的参数, 系统将取其默认值。例如, 上例中省略的参数 `target`, 在系统实际执行具体的检查时, 该参数就取被监控主机的 IP 地址。而省略的 `packets` 和 `size` 参数, 则取默认值 3 和 68 (指在 x86_64 平台执行 `fping` 命令)。而在关键字中, 如果省略的参数后边没有其他参数, 或者虽还有其他参数, 但是这些参数也被省略了, 则这个时候, 分隔参数的逗号(,)也可以省略。例如上例中, 如果省略这个关键字的最后一个参数 `timeout`, 则参数 `timeout` 前面的“,”号也可以省略, 即写成 `icmpping[,200,,]` 形式也是完全合法的。而如果监控项目的关键字是监控项目标识的一部分, 也即如果省略了某些参数, 则系统将无法区分不同的监控项目, 这些参数不能被省略。例如, 上面所给出的 `zabbix[wcache,history,total]`、`zabbix[wcache,values,not supported]`、`hrStorageUsedInBytes[/]` 等关键字, 它们的参数就不能省略。这是因为, 如果省略了这些关键字的参数, 系统将无法区分某个关键字是属于哪个监控项目的了。比如, 关键字“`hrStorageUsedInBytes[/]`”, 表示采集被监控主机系统中根分区已使用的空间大小数据, 如果省略了这个关键字中的参数“/”, 则系统就不知道该采集哪个分区的已使用空间大小数据了。

前面已经介绍过, 关键字的参数可以用引号引起来的字符串, 也可以是不用引号引起来的字符串。既然作为关键字参数的字符串, 既可以被引号引起来, 也可以不用引号引起来。那么, 同一个关键字的同一个参数, 用不用引号将其引起来有什么区别吗? 当一个关键字的参数用引号(包括单引号和双引号)引起来时, 则该参数就可以由任何 Unicode 编码的字符串组成。其中, 如果参数本身就包含单引号和双引号, 则参数本身的单引号和双引号需要分别用“\”和“”来进行转义。而当参数不用引号引起来时, 则参数名中不支持逗号(,)和右边中括号(])等字符。在实际的配置过程中, 除了参数名是由全数字字符所构成之外, 其他所有形式的关键字参数, 我们都建议尽量使用引号将其引起来, 以避免出现一些很莫名其妙的问题。

在 Zabbix 系统中, 监控项目关键字的参数还可以是数组。但是, 在监控项目关键字参数中使用的数组与其他许多高级编程语言中所定义的数组可能还有不同。在许多高级编程语言中所定义的数组, 都会有一个数组名(或者称为变量名), 而使用在监控项目关键字参数中的数组却没有数组名。在 Zabbix 系统中监控项目的关键字参数中, 只要是连续两个或两个以上的, 且

用中括号括起来的参数，系统都将其视为数组。例如在关键字 `zabbix[wcache,[history,total]]` 的参数列表中，就有数组参数 `[history,total]`，它与关键字 `zabbix[wcache,history,total]` 并不是同一个关键字。

在 Zabbix 系统预定义的关键字中，有某些关键字包含有 `encoding` 参数。这个参数是用来指定系统如何处理监控项目编码的，如果这个参数为空或省略，则系统将会使用 UTF8 编码来处理监控项目上所采集的监控数据。至于哪些关键字中包含有 `encoding` 参数，请参考本书附录 C 所列的内容。

4.1.2 Zabbix 系统中预定义的关键字

在前面章节中，我们提到过，在 Zabbix 系统的监控项目中，有一部分监控项目的关键字可以从系统预定义好的关键字中选择。而至于哪些关键字可以从系统预定义好的关键字中选择，哪些关键字又必须由我们手工输入，这跟监控项目的类型（与我们在前面章节所介绍的监控数据采集方法相对应）有关。具体来说，使用“通过 Zabbix 被监控设备代理采集监控数据”（包括主动和被动方式）、“简单检查”、“SNMP 协议陷入”、“Zabbix 系统内部数据采集”、“数据聚合”、“通过 SSH 协议采集监控数据”、“通过 Telnet 协议采集监控数据”和“通过 JMX 协议采集监控数据”等方法采集监控数据的监控项目都需要使用系统中预定义的关键字格式。这其中，使用“通过 Zabbix 被监控设备代理采集监控数据”（包括主动和被动方式）、“简单检查”、“SNMP 协议陷入”、“Zabbix 系统内部数据采集”、“数据聚合”等方法采集监控数据的监控项目，其关键字必须使用 Zabbix 系统中预定义的格式，否则系统就报监控项目不被支持的错误。而像使用“通过 SSH 协议采集监控数据”和“通过 Telnet 协议采集监控数据”等方法采集监控数据的监控项目，其关键字其实我们是可以自由输入的，只是有一个前提条件，那就是被监控主机上的 SSH 服务和 TELENT 服务所侦听的端口必须是对应服务的默认端口，否则 Zabbix 系统服务器端组件将无法连接到被监控主机的相应服务，也就谈不上在被监控主机上执行系统命令序列或脚本程序了，当然也就无法采集到需要的监控数据了。

或许读者还记得，在前面的介绍中提到过，如果要想通过 JMX 协议采集监控数据，则我们在编译安装 Zabbix 服务器端组件时，是需要一同编译安装 Java 应用程序网关的，且在系统正常工作时需要开启这个 Java 应用程序网关。所以，“通过 JMX 协议采集监控数据”的监控项目，其关键字形式也是由系统预定义好的，我们一般不可以随便更改，否则系统将无法识别出我们所配置的监控项目的关键字。

对于通过“简单检查”、“Zabbix 系统内部数据采集”和“SNMP 陷入”等方法采集监控数据的监控项目，其可以使用的关键字，我们已经在本书的第 2 章做过详细介绍，在此就不再复述了。而使用“通过 Zabbix 被监控设备代理采集监控数据”方法采集监控数据的监控项目，因其可选的关键字非常多，故我们将它们进行整理后，集中以附录的形式列在本书的最后，有需要的读者可参阅本书的附录 C。而其他系统预定义的关键字其形式和含义都比较简单，在这里，就拣几个这样的关键字简单地介绍一下。

“数据聚合”类监控项目的关键字形式为：`fun[para1, para2,item_fun,para3]`。其中，`fun` 为组聚合函数，可选的组聚合函数，在第 2 章详细介绍过。`para1` 为参与数据聚合的主机或主机组列表，用逗号（,）分隔；参数 `para2` 为需要被聚合的监控项目的关键字；而 `item_fun` 表示监控项目函数，可选的监控项目函数，我们在本书的第 2 章已经作过详细介绍；最后一个参数 `para3` 是时间参数。下面是几个数据聚合类监控项目关键字的例子。

例 1: `grpsum["MySQL Servers","vfs.fs.size[/,total]",last,0]`

功能: 求系统中, 主机组属于“MySQL Servers”的所有主机的根分区容量总和, 并且计算所依据的数据是, 系统最近一次采集到的“MySQL Server”主机组中所有主机的根分区容量。

例 2: `grpavg["Web Servers","system.cpu.load[,avg1]",last,0]`

功能: 计算系统中, 主机组属于“Web Servers”的所有主机的一分钟平均负载的平均值, 并将计算所得到的数据, 作为新监控项目所采集到的监控数据。计算时, 系统所依据的数据是, 系统最近一次采集到的“Web Servers”主机组中所有主机一分钟平均负载值。

例 3: `grpavg[["web1","web2","web3"],system.cpu.load,last,0]`

功能: 求“web1”、“web2”和“web3”三台主机负载的平均值, 以系统最新采集到的监控数据作为计算的依据, 并将计算的结果视为新监控项目所采集到的监控数据保存。

例 4: `grpavg["MySQL Servers",mysql.qps,avg,5m]`

功能: 统计属于“MySQL Servers”主机组的所有主机, 5 分钟内平均执行查询的次数, 并将计算结果视为新监控项目所采集到的监控数据。

注意: 1. 如果关键字中的第 3 个参数即监控项目函数, 使用的是 last 函数, 则该关键字中的第 4 个参数即时间参数, 将会被忽略。

2. 关键字中的第 4 个参数只能是表示时间的数字, 即该参数的数据前不接受#号。

3. 在进行数据聚合计算时, 系统将只取状态为“启用”的主机和监控项目所采集的监控数据, 非启用状态下的主机和监控项目, 即使其属于被聚合的对象, 它也不会参与数据聚合, 且系统不会给出任何提示。

前面已经介绍过, 对于通过 SSH 协议和 TELNET 协议采集监控数据的监控项目, 其关键字可以由我们自由地输入, 但是前提条件是, 被监控主机上所运行的 SSH 或 TELNET 服务侦听的端口必须是对应的默认端口。否则, 就需要使用系统中预定义的关键字。通过 SSH 协议采集监控数据类监控项目, 其系统预定义的关键字格式很简单, 格式为: `ssh.run[<unique short description>,<ip>,<port>,<encoding>]`。其中, 参数“unique short description”表示同一台被监控主机, 不同监控项目的唯一标识符, 其可以由我们自由命名, 只需遵守关键字参数的命名规范即可; 参数“ip”, 当然是指被监控主机的 IP 地址或者是其合法解析的 DNS 主机名; 参数“port”, 则是指被监控主机上所运行的 SSH 服务的端口号, 默认值是 22; 参数“encoding”, 则是指编码, 例如 GBK、UTF8 等。

通过 TELNET 协议采集监控数据类监控项目, 其关键字格式与通过 SSH 协议采集监控数据类监控项目的关键字格式类似, 它的关键字格式为: `telnet.run[<unique short description>,<ip>,<port>,<encoding>]`。参数“unique short description”表示同一个被监控主机上不同监控项目的唯一标识符, 其可以由我们自由命名, 只需遵守关键字参数的命名规范即可; 参数“ip”, 当然是指被监控主机的 IP 地址或者是其合法解析的 DNS 主机名; 参数“port”, 则是指被监控主机上所运行的 TELNET 服务的端口号, 默认值是 23; 参数“encoding”, 则是指编码, 例如 GBK、UTF8 等。

通过 JMX 协议采集监控数据类监控项目, 其关键字的格式同样也比较简单, 它的格式为: `jmx[<object name>,<attribute name>]`。其中, 参数“object name”表示所需要监控的对象名称;

参数“attribute name”表示属性名。在这两个参数中，对象名比较简单，就是在 Java 应用中定义的对象的名字字符串。而属性名则可能稍微有一点复杂，它的复杂性主要在于，不同对象下不同属性的返回值的情形比较多，且比较复杂。

当某个属性的返回值只是比较简单的一个字符串或整数时，则对应监控项目的关键字直接写成 `jmx[<object name>,<attribute name>]` 形式即可。例如关键字“`jmx["Catalina:type=GlobalRequestProcessor,name=http-8080",requestCount]`”，即表示 Tomcat 容器 8080 端口每 ns 所处理的请求的个数。在这个关键字中，参数“`Catalina:type=GlobalRequestProcessor,name=http-8080`”即为对象名称，而参数“`requestCount`”则表示该对象下请求数。但是，在实际应用中，我们经常会遇到某个属性的返回结果还是一个对象，而不是一个简单的字符串或数字。很显然，当某个属性的返回结果是一个对象时，我们在监控系统中无法直接应用这个对象。因此，这种具有对象性质的返回结果，如果不加以进一步处理对于我们监控某个应用是没有什么作用的。例如，在对象“`java.lang:type=MemoryPool,name=Perm Gen`”中有个“`Usage`”属性。当查询该属性的值时，系统将会返回一个对象性质的结果，其中包括“`max`”、“`used`”和“`committed`”等子属性的值。这个时候，就只需将这类监控项目关键字中属性参数进一步细化，就可以采集到我们所需要的单个子属性的数据了。比如，如果想采集 JVM 中永久内存的最大值，就只需将对应监控项目的关键字写成 `jmx["java.lang:type=MemoryPool,name=Perm Gen ",Usage.max]` 形式即可。相应的，如果要采集已使用的永久内存大小，则其所对应监控项目的关键字就应写成 `jmx["java.lang:type=MemoryPool,name=Perm Gen ",Usage.max]` 形式。

综上所述，当 JMX 对象的某个属性的返回结果中包含多个属性的值时，就只需使用点号(.) 在对应监控项目关键字中针对属性参数进一步细化，就可以获得单个属性的数据值。但是，使用点号进一步分隔一个属性和它的子属性，也会带来另外一个问题。如果某个属性的名字中本身就带点号，这个时候，当使用点号来分隔父子属性时，系统就将无法识别出应以哪个点号分隔父属性和子属性。比如，有一个对象的属性为“`fruits.apple.weight`”，此时，系统就将无法判别是将这个属性进一步分隔成 `fruits.apple` 主属性和 `weight` 子属性，还是分隔成 `fruits` 主属性和 `apple.weight` 子属性。为避免出现上述混淆，在监控项目关键字中，我们将属性名称中的点号用“`\`”来进行转义。如上面的属性，我们将其写成 `fruits\apple.weight` 形式，则 Zabbix 系统就可以识别出该属性的主属性为 `fruits.apple`，而子属性为 `weight`。另外，如果对象名称和属性名称中包含反斜杠(\)，则也需要用反斜杠(\)对它进行转义。

4.2 时间区间定义方法

在 Zabbix 系统中，有多个地方需要定义或使用时间区间，例如，在配置可变数据采集间隔时；在配置动作时，当选择触发条件为“时间周期”时，等等。定义时间区间的格式为：`d-d, hh:mm-hh:mm`。

其中，`d` 表示一个星期的第几天，例如，如果是星期一，则 `d` 为 1；星期二，则 `d` 为 2；……星期六，则 `d` 为 6；而如果是星期天，则 `d` 为 7。而 `hh` 为小时，其值范围为 00~24mm 则代表分钟，其值范围为 0~59。

当需要在某个配置项中配置多个时间区间时，则多个时间区间应以分号(;) 来隔开，其形式为 `d-d, hh:mm-hh:mm; d-d, hh:mm-hh:mm……`

如果在需要填写时间区间的地方没有输入任何内容，则系统将会使用默认的时间区间，即

相当于时间区间：01-07,00:00-24:00。

注意：确定了时间区间后，系统在实际使用时间区间时，将不包含大的时间端点。例如，如果定义了时间区间 09:00-18:00，则实际的时间范围是到 17:59:59 分为止，不包含 18:00 这个时间点。

以下是几个时间区间的例子，供读者学习时参考。

例 1：时间区间 1-5,9:00-18:00

例 1 表示的时间范围是，每个星期的星期一到星期五的上午 9 点到下午 6 点。这个时间范围一般也是我们的工作时间范围。所以，系统中预定义的“工作时间”区间，其范围就是上述的这个范围。当然，可以依次选择菜单“高级配置”→“常规”→“工作时间”，在“配置工作时间”页面上，根据我们实际的工作时间范围来修改这个“工作时间”的时间范围。“工作时间”区间，在 Zabbix 系统中有一个很有意思的应用。当打开监控项目的数据图时，我们可能很快就会发现，工作时间范围内的数据图是以白色底色来显示的，而非工作时间的数据图则是以灰色为底色来显示的。

例 2：时间区间 1-5,09:00-18:00;6-7,10:00-16:00

例 2 表示的时间范围是，每周的星期一到星期五的上午 9 点到下午 6 点，加上星期六和星期日的上午 10 点到下午 4 点。

4.3 历史数据和趋势数据

你或许还记得，我们在介绍如何创建监控项目时，介绍过在“配置项目”表单页面上有两个表单项“保留历史数据（天）”和“保留趋势数据（天）”，并且当时我们给出提示是，为减轻 Zabbix 系统数据库的压力，建议在配置监控项目时，应尽可能在“保留历史数据（天）”表单项里输入较小数字，比如说 7 天或 15 天等。那么，在 Zabbix 系统中，为何有“历史数据”和“趋势数据”两种数据呢？它们又都做什么用的呢？以及这两种类型的数据有什么联系和区别呢？下面，就针对这些问题谈谈 Zabbix 系统中的“历史数据”和“趋势数据”。

历史数据和趋势数据，是 Zabbix 系统存储所采取的监控数据的两种存储方式。它们既有区别又有联系。所谓历史数据是指，Zabbix 系统针对每个监控项目，在每次采集时所收集到的数据，这类数据保存在 Zabbix 系统数据库的历史表中。因为是每次 Zabbix 系统针对每个监控项目所采集到的数据都保存在历史表中，所以监控项目所配置的更新间隔越小，则在一定时间长度内保存到历史表中的数据就越多。如果每个监控项目的更新间隔都被配置成 30 秒，则在两个小时内，该监控项目在 Zabbix 数据库的历史表中就会有 240 条数据记录，一天就会有 2880 条记录。或许你认为，对于 MySQL 数据库来说，2880 条记录几乎可以忽略不计。是的，如果 Zabbix 系统只监控一台主机，且这台主机上只有一个监控项目，那么系统每天所产生的 2880 条记录确实不值得一提。但是，当监控系统监控的项目比较多时，这个数据量还是非常大的。比如说，监控系统监控了 1000 个监控项目，且每个监控项目的更新间隔都被配置成 30 秒，那么系统每天在历史表中就会产生 $2\,880 \times 1\,000 = 2\,880\,000$ 条记录，也即近 300 万条记录。而 1000 个监控项

目可以是多少台主机所拥有的呢？我们以 48 口的交换机为例，单以监控每台交换机的每个端口的流量来计算，则一台 48 口的交换机就有 96 个监控项目。所以，如果系统中所有的被监控主机都是这种 48 口的交换机，那么差不多 10 台这样的交换机就会有大约 1000 个监控项目。由此可见，如果监控主机的数量再稍多一点，或者更确切地说，我们监控的项目再稍微多点，则 Zabbix 系统每天在其数据库中产生的记录数是非常大的。因此，我们建议，如非必须，在配置监控项目时，应尽量减少历史数据的保留天数，以免给 Zabbix 系统数据库带来很大的压力。

而趋势数据则不同，对于相同的更新间隔和相同的监控项目数，系统产生的趋势数据的数量远远没有历史数据那么庞大。趋势数据的数据量之所以要远远小于历史数据的数据量，是由趋势数据的取值方式决定的。趋势数据取值方式是，它取对应监控项目的历史数据在一个小时内的平均值、最大值、最小值以及这一小时内该监控项目所采集到的监控数据的个数。因此，不管一个监控项目的更新间隔被设置成多少，它所对应的趋势数据在数据库中的记录都只有一条。更新间隔小，仅可能导致历史数据数据量的增大，而不会影响监控项目在趋势表里的记录条数。由此，你或许觉得趋势数据很不准确，还是更愿意保留较长时间的历史数据，以便查看较长时间的数据图。其实不然，这是因为在 Zabbix 系统数据库的趋势表中不但保留有监控项目一个小时内历史数据的最大值、最小值和平均值，而且还保存有这一个小时内对应监控项目所采集到的监控数据的个数。因此，在要求并不是很高的场景下，使用趋势数据所绘出的数据图的走势，与用历史数据绘出的数据图的走势差别不是很大。

不管是历史数据还是趋势数据，都会周期性地被 Zabbix 服务器端一种被称之为“管家（housekeeper）”的进程进行清理，该进程会周期性地删除过期的历史数据和趋势数据。也正是因为这个进程的存在，才使得 Zabbix 系统数据库中的数据量不会一直膨胀下去。而实际上，如果在保持 Zabbix 系统中被监控主机和监控项目不变，且不更改监控项目的更新间隔的情况下，Zabbix 系统数据库中的数据量在增长到一定的量后不会再增长，而是基本上保持不变。“管家”进程清理历史数据和趋势数据的频率，可以在 Zabbix 服务器端组件（或其代理组件）的配置文件 `zabbix_server.conf`（或 `zabbix_proxy.conf`）中进行设置，对应的配置项是 `HousekeepingFrequency`。

- 注意：1. 如果监控项目中的“保留历史数据（天）”配置项被设置成 0，则 Zabbix 系统数据库历史表中仅保留对应监控项目采集的最新一条监控数据，其他历史数据都将不会被保留。而且，引用该监控项目的触发器，其计算表达式中也只能使用该监控项目采集的最新监控数据。因此，此时如果要在触发器表达式中引用该监控项目，则使用 `max`、`avg`、`min` 等函数将会变得没有任何意义。
2. 如果监控项目的“保留趋势数据（天）”配置项被设置成 0，则该监控项目在系统数据库的趋势表中将不保留任何数据。

4.4 被监控设备代理组件的扩展

通过前面的介绍，我们已经知道，在 Zabbix 系统所支持的众多监控数据采集方法中，通过被监控设备代理组件来采集监控数据具有配置简单，数据采集稳定，所支持的平台比较多且不需要我们额外编写数据采集脚本或程序等优点。但是，通过这种方法采集监控数据也有一个缺

点,那就是,它所支持采集的监控数据的种类是相对固定的,是由 Zabbix 系统在开发的时候预定了好的。因此,很多用户认为,用这种监控数据采集方法很难满足自己对监控数据采集的个性化需求。果真是这样吗?其实不然,Zabbix 系统的被监控设备代理组件,为我们提供了一种被 Zabbix 官方称之为“用户参数(User Parameters)”的功能。通过这个功能,可以很方便地扩展被监控设备代理组件所能采集的监控数据的种类。如前文所说,这个功能在 Zabbix 系统官方手册被称之为“用户参数(User Parameters)”,在这里我们为了避免读者的误解和迷惑,将之称为对被监控设备代理组件的扩展,简称为代理组件扩展。

对被监控设备代理组件进行扩展其实非常简单,只需使用任何一款编辑器,将其配置文件 `zabbix_agentd.conf` (该配置文件在 Zabbix 安装目录的 `etc` 目录下)打开。然后,在配置文件中找到 `UserParameter` 配置项,修改这个配置项的内容。该配置项的格式为: `UserParameter=<key>, <command>`。其中, `<key>` 就是被扩展后的关键字;而 `<command>` 则是为获取监控数据所需要执行的命令序列或脚本程序。例如,如果将这个配置项的内容修改为: `UserParameter=get_nginxnum,/bin/ps -ef |grep "nginx" |/bin/grep -v "grep" |/usr/bin/wc -l`,并重启 `agentd` 进程,则就已经成功地扩展了被监控代理组件,使其可以获取本机上所运行的 `nginx` 进程的个数。修改好配置文件后,保存并重启 `zabbix_agentd` 进程,就可以通过 Web 前端组件,像添加普通监控项目一样,在 Zabbix 系统中添加新扩展的监控项目。只需注意的是,所添加监控项目的关键字需与我们在配置选项“`UserParameter`”中指定的 `key` 值保持一致。例如,这里添加的监控项目的关键字就是“`get_nginxnum`”。同时,监控项目类型选择“Zabbix agent”和“Zabbix agent (主动方式)”两者中之一均可。需要注意的是,在 `UserParameter` 配置项中指定的命令序列或脚本程序,被成功执行后,其返回的结果数据大小不应大于 64KB (Zabbix 2.0.5 以后版本,系统可以接受不大于 512KB 的数据)。

或许你会说,按照这里介绍的方法,即使能对被监控设备代理进行扩展,最多也只能多监控一个自定义监控项目而已。是的,如果按照我们上面所说的方法,的确只能多监控一个监控项目。但是,幸运的是,当通过 `UserParameter` 配置项来扩展被监控设备代理时,可以在关键字 (`key`) 中使用参数。例如,如果将 `UserParameter` 配置项的内容修改成 `UserParameter=get_nginxnum[*],/bin/ps -ef |grep $1 |/bin/grep -v "grep" |/usr/bin/wc -l`,则系统就可以按照我们在监控项目关键字参数中所指定的内容,来获取系统中指定进程的进程数了。比如说,我们在 Web 前端页面上配置监控项目时,将监控项目的关键字写成 `get_nginxnum[nginx]`,则对应监控项目所采集的监控数据就是,被监控主机上运行的 `nginx` 进程数;而如果将该监控项目的关键字写成 `get_nginxnum[zabbix]` 形式,则对应监控项目上所采集的监控数据就是被监控主机上运行的 `zabbix` 进程数。因此,通过这种方法,就可以自定义大量监控项目了。

上例中, `UserParameter` 配置项中的“`get_nginxnum[*]`”表示定义一个能带参数的关键字。在 `UserParameter` 配置项中配置关键字时,关键字的名称可以随便取,只需符合关键字的命名规范即可,但是其形式是固定的,即只能定义成“关键字名称[*]”的形式。其中,中括号里的内容表示的是关键字的参数。而关键字中所定义的参数是通过 `$1`、`$2`、`$3`、……、`$9` 等变量形式传递给配置项中指定的命令序列和脚本程序的。例如,我们上面所举的例子“`/bin/ps -ef |grep $1 |/bin/grep -v "grep" |/usr/bin/wc -l`”中的位置变量 `$1`。通过这种方式可传递的关键字参数最多可以达 9 个。下面我们再给出几个扩展被监控设备代理组件的例子,在下面的例子中,就不再一步步地说明如何修改被监控设备代理配置文件中的 `UserParameter` 配置项了,你只需将这个配置项的内容修改为我们例子中的内容,然后重启 `zabbix_agentd` 进程即可。

例 1: `UserParameter=wc[*],grep -c "$2" $1`

功能: 统计并返回指定文件中指定字符串的行数。

实例: `wc[/opt/nginx/logs/access.log,404]`

含义: 统计网站访问日志中出现 404 请求的次数。

例 2: `UserParameter=mysql.status[*],echo "show status" |mysql -u$1 -p$2 |grep "$3" |awk -F " " '{print $2}'`

功能: 获取 mysql 数据库的状态信息。

实例 1: `mysql.status[root,password,Threads_connected]`

含义: 获取 MySQL 数据库当前被连接的线程数。其中, 关键字中 root 即为连接数据库的用户名, password 为连接数据库的密码, 而 Threads_connected 则为要匹配的关键字。

实例 2: `mysql.status[root,password,Slow_queries]`

含义: 获取 MySQL 数据库执行慢查询的次数。

注意: 1. 因为 \$1、\$2 等变量名被关键字中的参数使用了, 所以在命令序列中, 如果需要使用这种变量名, 则需要在变量名前面再加一个美元符号。例如我们在 UserParameter 配置项中所配置的命令部分内容为 `/bin/echo "aaaa:bbb" |/bin/awk -F ":" '{print $$2}'`, 该命令在执行时, 就会打出第二个字段的内容 “bbb”。

2. 如果配置文件 `zabbix_agentd.conf` 中的配置项 `UnsafeUserParameters` 被设置成 1, 则在关键字中不可包含如下字符: \、'、”、`、*、?、[、]、{、}、~、\$、!、&、;、(、)、<、>、|、#、@。

3. 通过被监控设备代理扩展的方法采集监控数据, 其返回的结果可以是文本、字符串和日志类型的信息。系统可以接受由空格组成的字符串, 但是, 如果命令序列或脚本程序没有返回任何字符, 则被监控设备代理程序将返回 `ZBX_NOTSUPPORTED` 字符串, 即表示对应的监控项目不被支持。

4.5 动态索引

在前面介绍通过 SNMP 协议采集监控数据时曾提到, Zabbix 服务器端 (或其代理端) 会使用监控项目上配置的 OID 值, 定期通过 SNMP 协议到被监控主机或其代理上查询要采集的监控数据。这样, 就需要为每个通过 SNMP 协议采集监控数据的监控项目配置 OID 值。而如果为系统中每个监控项目都单独配置属于它们自己的 OID 值, 这无疑是一项工作量非常巨大的工作, 而且也不易管理, 容易出错。故此, 在前面章节中提到过解决这个问题的几种方法, 其中之一就是使用动态索引的方法。其实, 使用动态索引除了能减小日常维护 Zabbix 系统的工作量, 减少因为手工操作而出现的差错外, 在某些场景下, 使用动态索引的方法获取对象的 OID 值估计也是最高效、最便捷的手段。例如, 对于服务器, 它们所安装的硬盘数量和磁盘被划分的分区的个数, 很可能每台服务器都不一样。这个时候, 如果针对每台服务器逐个配置与磁盘相关的监控项目, 那么, 当所要监控的服务器的数量比较大时, 那工作量将是非常可观的。所以, 在这种场景下, 使用动态索引, 结合 Zabbix 系统所提供的自动发现功能 (何为 Zabbix 系统的自

动发现功能，以及如何使用它，将在后续章节中作详细介绍）将是最好的方法。那么，什么是动态索引，以及我们如何在 Zabbix 系统中使用动态索引呢？本节，我们将详细介绍这方面的相关知识。

4.5.1 动态索引介绍

在具体介绍动态索引之前，让我们先使用 net-snmp 软件包所提供的 snmpwalk 命令工具，来抓取一台网络交换机的网络端口的详细信息。

```
shell> snmpwalk -v2c -c snmp@domain.com -O n 192.168.5.9 IF-MIB::ifDescr
```

系统将返回如下信息内容：

```
.1.3.6.1.2.1.2.2.1.2.1 = STRING: Vlan1
.1.3.6.1.2.1.2.2.1.2.10 = STRING: Vlan10
.1.3.6.1.2.1.2.2.1.2.5001 = STRING: Port-channel1
.1.3.6.1.2.1.2.2.1.2.5002 = STRING: Port-channel2
.1.3.6.1.2.1.2.2.1.2.5003 = STRING: Port-channel3
.1.3.6.1.2.1.2.2.1.2.5004 = STRING: Port-channel4
.1.3.6.1.2.1.2.2.1.2.10101 = STRING: GigabitEthernet0/1
.1.3.6.1.2.1.2.2.1.2.10102 = STRING: GigabitEthernet0/2
.1.3.6.1.2.1.2.2.1.2.10103 = STRING: GigabitEthernet0/3
.1.3.6.1.2.1.2.2.1.2.10104 = STRING: GigabitEthernet0/4
.1.3.6.1.2.1.2.2.1.2.10105 = STRING: GigabitEthernet0/5
.1.3.6.1.2.1.2.2.1.2.10106 = STRING: GigabitEthernet0/6
...
```

从返回结果不难看出，每一行代表对应交换机的一个网络端口信息。其中，等号左边的数据是对应端口的 OID 值，而等号右边的数据则是交换机中所配置的对应该端口的描述信息。首先来研究一下该交换机端口的 OID 值，在所有这些端口的 OID 值中，“.1.3.6.1.2.1.2.2.1.2.”部分是相同的，而改变的只是 OID 值中以点号（.）分隔的最后一部分。这里获取的是网络交换机端口的描述信息，而如果获取该交换机各个端口的流量信息，是不是也呈现出这样的规律呢？下面即为我们在抓取同一台网络交换机的各个端口出口流量信息时，系统给我们的返回结果。

```
shell> snmpwalk -v2c -c snmp@domain.com -O n 192.168.5.9 IF-MIB::ifInOctets
```

```
.1.3.6.1.2.1.2.2.1.10.1 = Counter32: 0
.1.3.6.1.2.1.2.2.1.10.10 = Counter32: 275732995
.1.3.6.1.2.1.2.2.1.10.5001 = Counter32: 28832717
.1.3.6.1.2.1.2.2.1.10.5002 = Counter32: 61438686
.1.3.6.1.2.1.2.2.1.10.5003 = Counter32: 24303332
.1.3.6.1.2.1.2.2.1.10.5004 = Counter32: 85189929
.1.3.6.1.2.1.2.2.1.10.10101 = Counter32: 3006843847
.1.3.6.1.2.1.2.2.1.10.10102 = Counter32: 586787460
.1.3.6.1.2.1.2.2.1.10.10103 = Counter32: 1770203309
.1.3.6.1.2.1.2.2.1.10.10104 = Counter32: 2318412
.1.3.6.1.2.1.2.2.1.10.10105 = Counter32: 4293900
.1.3.6.1.2.1.2.2.1.10.10106 = Counter32: 57025122
...
```

从上述系统的返回结果不难看出，该网络交换机不同网络端口的 OID 值，变化的也只是其用点号分隔的最后一部分，OID 值的其他部分内容，对于该台网络交换的每一个端口都是相同的。

既然这样，那么可不可以这么认为，对网络交换机来说，其不同网络端口所对应的 OID 值变化的部分，其实就是同一类对象不同个体在被监控设备的管理信息库（MIB）中，用于区分同类对象其他个体的系统标识？也即，不管要查询同一类对象的哪一个具体个体的属性值，只

需修改该个体 OID 值的最后一部分，其他部分内容保持不变，就可以查询到不同个体的属性值了？例如，在我们上面所抓取网络交换机不同网络端口流量值的返回结果中，OID 值为.1.3.6.1.2.1.2.2.1.10.1 的行，其所对应的流量值其实就是 Vlan1 端口的流量；而 OID 值为.1.3.6.1.2.1.2.2.1.10.10101 的行，其所对应的流量值其实就是端口 GigabitEthernet0/1 的流量值。是的，你所思考的一点都没有错，交换机的不同端口流量值和它的 OID 值之间的关系就是这样的。

既然如此，那么，对于监控项目数量会随着系统实际配置情况而改变的监控系统，其配置就比较好处理了。这是因为，既然有这么一个规则，那么我们在配置这类监控项目时，就不需要知道该类监控项目在实际被监控设备上到底有多少个，而只需配置一定的规则，先将被监控设备上已存在的某类对象的个数及其描述信息抓取出来。然后，根据抓取结果，从中分离出对该类对象 OID 值中变化和非变化的部分。最后，根据上述规则，逐个调整该类对象 OID 值中变化的部分，并实施监控数据的采集。这个功能，在 Zabbix 系统中被称之为“低级自动发现（Low-level Discovery, LLD）”功能。当然，如果不想使用 Zabbix 系统中的低级自动发现功能，你也可以在配置监控项目时，将监控项目所需的 OID 值指定为某个具体对象的 ID 值。这样，系统就只会采集该具体对象所对应的数据了。这里还是以抓取网络交换机不同网络端口流量值为例，详细介绍在 Zabbix 系统中使用动态索引功能的方法和原理。

首先来介绍如果不使用 Zabbix 系统中的低级自动发现功能，该如何利用动态索引的方法采集监控数据。如果要在不使用低级自动发现功能的情况下使用动态索引功能，则监控项目的 OID 值需要填写成<OID of data>["index",<base OID of index>,<string to search for>"] 格式。在该格式的 OID 值中，各参数的含义如下。

- ❑ **OID of Data:** 监控对象属性的主 OID 值。它用来指定，系统将采集对象的哪个属性的数据。例如，系统采集网络交换机网络端口的流出流量，还是采集网络交换机网络端口的流入流量或者是网络端口的传输速率等。该参数的值，只能从 Zabbix 系统中所定义的特殊 OID 值集合中选择，不可以自己随意填写。至于 Zabbix 系统中定义了哪些特殊 OID 值，我们将在下一节中详细介绍。
- ❑ **index:** 索引。用于指定 Zabbix 系统如何处理监控项目的 OID 值。目前这个参数位置只能填写 index。
- ❑ **base OID of index:** 指定用于查找索引（index）的基础 OID 值。什么含义呢？具体的意思就是，Zabbix 系统将从这个参数所指定的 OID 值开始查找其下的所有子叶中参数“string to search for”指定的字符串对象。
- ❑ **string to search for:** 指定要匹配的字符串。该参数的内容是大小写敏感的。

在不使用 Zabbix 系统低级自动发现功能时，动态索引功能的工作流程是，系统从监控项目 OID 值的“base OID of index”参数指定的 OID 值，开始搜索被监控主机上 MIB 库里的信息，并用参数“String to search for”中所指定的内容匹配搜索到的结果。如果匹配到了记录，则系统从匹配到的记录中分离出要被监控对象 OID 值的动态部分（也即对象 OID 值的索引）。然后，系统将这个索引值添加到监控项目上主 OID 值后面，形成 OID of Data.index 形式的新 OID 值，最后，系统再次使用新的 OID 值查询被监控主机上的 MIB 库，并从中采集需要的监控数据。或许这么介绍，还是很抽象，你可能还是没有看明白系统是怎么结合动态索引工作的。下面就举个例子，来说明 Zabbix 系统是如何通过动态索引的方法，获取网络交换机上 GigabitEthernet0/1

网络端口的网络流出流量的。

如果要采集到网络交换机上 GigabitEthernet0/1 网络端口的网络流出流量值, 则该网络端口所对应的监控项目的 OID 值, 就要被配置成 `ifOutOctets["index","ifDescr","GigabitEthernet0/1"]`。首先, Zabbix 系统从 “ifDescr” 这个 OID 值 (这个 OID 值是 Zabbix 系统所定义的特殊 OID 值全集中的一个, 它的数字表现形式为: .1.3.6.1.2.1.2.2.1.2) 开始搜索被监控主机上的 MIB 信息库。返回的结果与我们在前面使用 `snmpwalk` 命令查询时返回的结果类似。然后, Zabbix 系统在系统返回结果的 “值” 部分 (即每条记录等号右边的部分) 匹配字符串 “GigabitEthernet0/1”, 并将会成功匹配到记录 `.1.3.6.1.2.1.2.2.1.2.10101 = STRING: GigabitEthernet0/1`。接下来, 系统从匹配到的记录中, 分离出 OID 值的索引部分, 即 10101。之后, Zabbix 系统将从记录中分离出来的索引值 “10101”, 添加到主 OID 值 `ifOutOctets` 的后面, 形成新的 OID 值 `ifOutOctets.10101`。如上文所述, 主 OID 值也是 Zabbix 系统中定义的特殊 OID 值, 如果将其书写成数字形式, 则该 OID 值的数字形式为: .1.3.6.1.2.1.2.2.1.10。这样, 添加上索引后, 所形成的新的 OID 值的数字表达形式为: .1.3.6.1.2.1.2.2.1.10.10101。拿着这个 OID 值, 在 Zabbix 服务器上执行 `snmpwalk -v2c -c snmp@domain.com -O n 192.168.5.9 IF-MIB::ifInOctets` 命令, 并在其成功执行后的返回结果中找找看。我们很快就可以找到, 记录 “.1.3.6.1.2.1.2.2.1.10.10101 = Counter32: 3006843847” 所对应的 OID 值, 与 Zabbix 系统通过动态索引的方法得到的 OID 值是一样的。因此, 最后当 Zabbix 系统使用新的 OID 值查询指定网络端口的网络流出量时, 其所查询到的数据就是网络交换机端口 GigabitEthernet0/1 网络流出量值。

上面介绍了在不使用低级自动发现功能时, Zabbix 系统是如何使用动态索引的方法采集监控项目的监控数据的。如果使用低级自动发现功能, 则其工作流程与不使用低级自动发现功能的工作流程是类似的。只是, 在使用低级自动发现功能时, 因为不是查找某个特定的对象 (例如, 上例中网络交换机的特定的端口 GigabitEthernet0/1), 所以, Zabbix 系统无须通过字符串匹配的方法, 来查找和定位特定对象的索引值。而是将所发现的所有同类型对象统统都 (如果配置有正则表达式, 则可以过滤掉某些无须监控的对象) 视为要被监控的对象, 并在系统中自动添加与所发现对象相对应的监控项目。所以, 从这个角度来说, 如果使用低级自动发现功能, 则使用动态索引的方法采集监控数据的流程还要简单一些。

4.5.2 特殊 OID 值

在前一节中提到过, Zabbix 系统定义了一些特殊 OID 值。说它们是特殊的 OID 值或许也并不太准确, 因为它们并不是 Zabbix 系统向被监控主机上 MIB 库里添加的新的 OID 值, 而只是 Zabbix 系统对某些通用的经常使用的 OID 值, 在 Zabbix 系统内部重新定义的一种简写形式。换句话说, 也可以将这些所谓特殊的 OID 值理解为, Zabbix 系统对某些通用的经常使用的 OID 值, 在 Zabbix 系统中给它们重新取了一个名字或标识符。当在 Zabbix 系统中使用这些所谓特殊的 OID 值时, 我们无须使用它们本来的值, 而只需使用它们被简化的名称即可, Zabbix 系统会自动将简化名称转化为它们本来的值。表 4-1 列出了 Zabbix 系统中定义的特殊 OID 值及它们的含义描述。

表 4-1 特殊OID值列表

特殊OID	标准OID	描述
ifIndex	.1.3.6.1.2.1.2.2.1.1	该OID及其下叶子记录的是主机中网卡，也包括网络交换机各端口的索引值，即如我们上一节例子中所述的10101值。该OID值在每台主机上，不同的网络接口中都是唯一的
ifDescr	1.3.6.1.2.1.2.2.1.2	该OID值所对应的是主机（也包括网络交换机等网络设备，下同）中各网络接口的字符串名称。如我们上一节所举例子中的“GigabitEthernet0/1”。该名称可能包含设备制造商的名称、设备名称和网络接口的版本信息等
ifType	1.3.6.1.2.1.2.2.1.3	该OID及其叶子所对应的值表示的是，主机中各个网络接口的类型。例如接口是以太网接口还是回环接口或者是虚拟网络接口等
ifMtu	1.3.6.1.2.1.2.2.1.4	该OID及其叶子所对应的值表示的是，主机中各个网络接口的最大传输单元（Maximum Transmission Unit, MTU）值。即网络接口上单次接收或发送的最大数据包大小，它的单位是字节
ifSpeed	1.3.6.1.2.1.2.2.1.5	该OID及其叶子所对应的值表示的是，主机中各个网络接口当前的带宽，单位是比特每秒
ifPhysAddress	1.3.6.1.2.1.2.2.1.6	该OID及其叶子所对应的值表示的是，主机中各个网络接口的物理地址，即MAC地址
ifAdminStatus	1.3.6.1.2.1.2.2.1.7	该OID及其叶子所对应的值表示的是，主机中各个网络接口的管理状态，即对应的网络接口是否被网络管理员给禁用掉了。如果已经被禁用，则其对应状态值为down，否则其对应状态值为up
ifOperStatus	1.3.6.1.2.1.2.2.1.8	该OID及其叶子所对应的值表示的是，主机中各个网络接口的操作状态，即对应的网络接口是否正常连接网络。以我们的经验，如果网络接口的管理状态为up，而操作状态为down，一般很可能是网线没有插好
ifInOctets	1.3.6.1.2.1.2.2.1.10	该OID及其叶子所对应的值表示的是，主机中各个网络接口所接收到的字节数。我们通过获取这个OID值对应对象的属性值，可以获取网络接口的网络流入流量
ifInUcastPkts	1.3.6.1.2.1.2.2.1.11	该OID及其叶子所对应的值表示的是，主机中各个网络接口所接收并转发到上层协议处理的单播数据包数
ifInNUcastPkts	1.3.6.1.2.1.2.2.1.12	该OID及其叶子所对应的值表示的是，主机中各个网络接口所接收并转到上层协议处理的非单播数据包数，比如网络中的广播包等
ifInDiscards	1.3.6.1.2.1.2.2.1.13	该OID及其叶子所对应的值表示的是，主机中各个网络接口接收到了，但是被丢弃的数据包数
ifInErrors	1.3.6.1.2.1.2.2.1.14	该OID及其叶子所对应的值表示的是，主机中各个网络接口所接收到的错误数据包数

续表

特殊OID	标准OID	描述
ifInUnknownPkts	1.3.6.1.2.1.2.2.1.15	该OID及其叶子所对应的值表示的是，主机中各个网络接口所接收到的未知协议的数据包数
ifOutOctets	1.3.6.1.2.1.2.2.1.16	该OID及其叶子所对应的值表示的是，主机中各个网络接口所发送的数据字节数。我们通过获取这个OID值对象的属性值，可以获取网络接口的网络流出流量
ifOutUcastPkts	1.3.6.1.2.1.2.2.1.17	该OID及其叶子所对应的值表示的是，主机中各个网络接口被高层协议要求发送的非广播和组播包的数量，包括丢弃或未发送成功的数据包的数量
ifOutNUcastPkts	1.3.6.1.2.1.2.2.1.18	该OID及其叶子所对应的值表示的是，主机中各个网络接口被高层协议要求发送的广播和组播包的数量，包括丢弃或未发送成功的数据包的数量
ifOutDiscards	1.3.6.1.2.1.2.2.1.19	该OID及其叶子所对应的值表示的是，主机中各个网络接口被高层协议要求发送的，但是最终被丢弃的数据包的数量
ifOutErrors	1.3.6.1.2.1.2.2.1.20	该OID及其叶子所对应的值表示的是，主机中各个网络接口被高层协议要求发送的，但是最终发送出错的数据包数量
ifOutQLen	1.3.6.1.2.1.2.2.1.21	该OID及其叶子所对应的值表示的是，主机中各个网络接口发送数据队列长度

4.6 事件和事件源

在前面章节中介绍过，如何通过 Zabbix 系统的 Web 前端组件来查看系统中当前所发生的事件信息。同时，也提到过，在 Zabbix 系统中，事件消息有多个来源，例如触发器状态的改变、系统自动发现了某台主机或某个服务等，系统都会产生相应类型的事件信息。那么，在 Zabbix 系统中，事件有什么意义和作用呢？以及在 Zabbix 系统中，当出现哪些情况或场景时，系统就将其视为事件，并产生相应事件信息呢？本节就针对事件的这些问题进行介绍和讨论。

作为一款优秀的开源监控系统，我们当然希望它不但能够准确无误地采集所配置监控项目的监控数据，并按我们的要求绘制出数据图和图表。而且，我们还希望通过它能够查看和了解，系统监控的项目在之前的某段历史时间内，其状态是否发生过改变，以及状态改变发生的时间等信息。同时，对于自动发现功能（注意，这里所说的自动发现功能与前一节中所说的低级自动发现功能不是一回事），也希望能够通过它查看在历史的某段时间内，它为我们发现的主机或服务情况，以及被发现的主机或服务丢失了的情况。对于上面所述的这些情形，在 Zabbix 系统中都将其视为事件。并将其所发生的时间、详细信息等内容记录在系统数据库中。在需要时，可以通过 Web 前端组件的“状态统计”→“事件”菜单项来查阅这些信息。对于 Zabbix 系统中的事件，我们不仅通过它来了解我们所监控的项目和主机，在历史的某段时间内到底发生了哪些事件，以及发生这些事件的详细历史信息。更重要的是，在 Zabbix 系统中，事件是“动作”的驱动源。换句话说，只有系统中发生了新事件，才有可能触发系统中所配置的某个“动作”。

而只有至少一个“动作”被触发了，系统才有可能将报警信息发送到指定人员那里，或者执行指定的远程命令（动作步骤升级所发的报警信息或所执行的远程命令除外。因为在这种情况下，系统发送报警信息或执行远程命令并不是因为系统中有新的事件产生）。因此，Zabbix 系统中是否发生了新的事件，是系统是否会发送新的报警信息或执行远程命令的重要一环。当然，当系统中有一个新的事件发生时，是否有至少一个“动作”被触发，还跟系统中所配置动作的“触发条件”有关。而动作即使被触发了，系统是否会发送报警信息或执行远程命令，还跟动作的“行为”配置有关。

上面介绍了 Zabbix 系统中事件的作用和意义，接下来介绍在 Zabbix 系统中，在哪些情况或场景下，系统会产生新事件。

4.6.1 触发器类事件 (Trigger events)

这种系统事件来源很好理解，即只要系统中所配置的触发器的状态发生了改变，不管是它的状态从“正常(OK)”状态转变为“问题(PROBLEM)”状态，还是从“问题(PROBLEM)”状态转变为“正常(OK)”状态，系统都会生成相应的新事件。在实际 Zabbix 系统维护过程中，这类事件应该是我们接触到的最多的一类事件。或许你会问，当触发器的状态从“正常(OK)”状态转变为“问题(PROBLEM)”状态时，系统会产生一个新的事件，从而触发“动作”，并给我们发送报警信息或执行远程命令，这个很好理解。可为什么当触发器的状态从“问题(PROBLEM)”状态转变为“正常(OK)”状态时，系统也会生成一个新事件呢？其实，对于这个问题，我们不难理解。在实际监控系统中，应该配置有很多这样的监控项目：该类项目可能在某个时间段发生了“故障”，但是过了一段时间后，不需要任何外力（也或许是通过自动执行的脚本任务来完成的），故障就会自动地被排除。这个时候，我们当然想知道，该监控项目的故障是在何时被自动排除的，以及“故障”持续了多长时间等信息。因此，在这种情况下就需要查看故障的恢复事件信息。更重要的是，你或许还记得，我们在前面章节中介绍如何配置“动作”时，在“动作”选项卡的表单页面上有一项“恢复消息”表单项。当时介绍说，如果勾选了这个复选框，则相应故障在被排除后，系统就会向相关人员发送一条故障恢复的信息。因此，如果没有故障恢复事件，那么当时我们配置的“恢复消息”这个表单项就将不起任何作用。因为系统接收不到事件信息，也就无法触发动作，更不可能给相关人员发送恢复信息了。

对于触发器类事件，一旦事件发生，系统就会详细地记录相关触发器状态改变的信息，包括触发器状态改变发生的时间、是哪台主机上的哪个触发器的状态发生了改变、状态发生改变后对应触发器新的状态等信息等。

4.6.2 自动发现类事件 (Discovery events)

前面提到过，Zabbix 系统中有一种被称之为“自动发现(Discovery)”（注意，与前一节所介绍的低级自动发现功能不是同一种功能）的功能。它主要是根据所配置的规则，使用一定的方法，对指定网段范围内的所有 IP 地址进行扫描，以发现网络中是否有“存活”的主机或者某类服务。使用自动发现功能，可以大大地减小日常维护监控系统的工作量，减少人为差错的几率。特别是在大型网络监控中，使用自动发现功能不但可以让系统自动地添加被发现的主机或监控项目，而且也可以使用自动发现功能来验证或找出监控配置上的错误，比如，该监控的主机没有在系统中配置，该监控的项目漏监控了，等等。

具体来说，自动发现类事件会在表 4-2 所列的情形下产生。

表 4-2 产生自动发现类事件情形列表

事 件	何时生成对应事件
服务可用	在自动发现功能每发现一个新的可用服务时，系统会产生自动发现类新事件
服务不可用	是指在自动发现功能未能在某台主机上发现到指定的服务时，系统会产生自动发现类事件。需要注意的是，“服务不可用”事件与下面将要说明的“服务丢失”类事件是不相同的。服务不可用类事件，是指自动发现功能从未在某台主机上发现指定的服务时，系统所产生的事件。而“服务丢失”类事件则不同，它是指系统通过自动发现功能发现了某个服务，但是后来在下一次“发现”时，又发现该服务不可用。此时系统会产生“服务丢失”类事件
主机可用	当发现某台主机上至少有一个可用服务时，系统会产生这类新事件
主机不可用	当某台主机上与自动发现方法所对应的服务不可用时，系统就会产生该类新事件
发现服务	当某个服务从不可用状态转变为可用状态，或者系统首次通过自动发现功能发现了一个新服务时，系统会产生该类型新事件
服务丢失	是指系统之前通过自动发现功能发现了某个服务，后来又发现主机上该服务不可用，系统会产生该类型新事件
发现主机	当某个主机从不可用状态转变为可用状态，或者系统首次通过自动发现功能发现了一台新主机时，系统产生该类型新事件
主机丢失	是指系统之前通过自动发现功能发现了某台主机，后来又发现该主机不可用时，系统会产生该类型新事件

4.6.3 被监控设备代理自动注册类事件（Active agent auto-discovery events）

在具体介绍被监控设备代理自动注册类事件之前，我们先来介绍一下什么是被监控设备代理自动注册。Zabbix 系统的被监控设备代理自动注册功能，是一种与自动发现功能相类似的功能，通过它可以将之前没有在监控系统中配置的主机自动地添加到监控系统中，并正常地实施监控数据采集和报警等。但是，它与 Zabbix 系统的自动发现功能不同之处在于，自动发现功能是 Zabbix 服务器端组件根据我们所配置的自动发现规则和发现方法，扫描指定范围内的每个 IP 地址，并判断是否存在 IP 地址属于指定范围内的存活主机。如果系统发现了 IP 地址属于指定范围的主机，且该主机在 Zabbix 系统中还没有被添加和配置，则系统会自动产生一条自动发现类事件。该事件可以驱动自动发现类动作（如果系统中配置了这种类型的动作），以完成某种行为，例如给指定的用户发送自动发现类信息，或者自动将被发现的主机添加到系统中，并实施监控数据的采集与报警等。所以，从监控模式上来看，它属于一种被动监控模式。即上述操作是由 Zabbix 服务器（或其代理）端完成的，被监控主机不会主动地将自己“存活”的消息告知 Zabbix 服务器。而被监控设备代理自动注册过程的工作模式则完全不同。当监控网络中新增加了一台原来没有在监控系统中配置的主机，且在该主机上启动了被监控设备代理组件，同时被监控设备代理组件被配置成主动模式（其配置很简单，只需将其配置文件 `zabbix_agentd.conf` 中 `ServerActive` 配置项的内容修改为 Zabbix 服务器或 Zabbix 服务器代理的 IP 地址或合法的 DNS 主机名，并重启 `zabbix_agentd` 服务即可）时，则只要被监控设备代理组件一启动，便会主动向 Zabbix 服务器端（或其代理端）发送自身“存活”的消息。而 Zabbix 服务器端接收到该消息后，便会在系统内部产生一条被监控设备代理自动注册事件，该事件会驱动自动注册类动作。动作

被触发后会作出相关的行为操作，比如给相关人员发送被监控设备代理自动注册信息，或者自动在系统中添加对应主机，并将相应主机关联到指定的模板或者将对应主机添加到指定的主机组中等。一旦新发现的主机被添加到系统中，其也就和通过其他方法添加到系统中的主机一样，可以被系统正常地采集监控数据，发送监控报警信息等。

从上面的叙述中，不难看出，被监控设备代理自动注册的工作模式是一种主动工作模式，它不需要 Zabbix 服务器端组件定期检测主机是否存活，而是被监控主机一旦存活，便会将自己的存活消息主动地告诉 Zabbix 服务器。需要说明的是，虽然通过被监控设备代理自动注册功能添加到系统中的主机，是以主动的方式添加到系统中的。但是，只要该主机被成功地添加到了系统中，则对于其上监控项目是否通过主动方式采集监控数据不作要求。换句话说，只要主机被添加到系统中，则所有属于它的监控项目既可以通过被动方式采集监控数据，也可以通过主动方式采集监控数据。

通过被监控设备代理自动注册功能，自动地向监控系统中添加被监控主机，这非常适合于在云系统环境中应用。我们知道，云系统一个非常大的优势就是，软硬件资源可以实现按需动态分配。换句话说，云系统中的云节点是动态的、实时变化的。当有需求时，云系统就会自动添加新的节点，而当资源过剩时，云系统又能及时回收过剩软硬件资源。在这种情况下，如果完全依靠手工来维护实时变化的云节点的监控，不但费时费力，而且也很难跟上实时变化的节奏。而同时，云系统中的不同云节点，往往分布在跨区域的很大地理范围内。在这种情况下，如果使用自动发现功能，自动将云节点动态地添加到监控系统中，不但非常消耗 Zabbix 系统软硬件资源，而且因为云节点一般分布的区域非常广，因此通过这种方法将非常难于实现。但是，通过被监控设备代理自动注册功能，自动添加被监控主机方法则不同。因为，这种方法是被监控主机主动联系 Zabbix 服务器，所以它不需要 Zabbix 服务器周期性地作大范围主机存活状态扫描。而通过被监控设备代理自动注册这种方法，只有存活的主机才会定点向 Zabbix 服务器端主动发送信息。所以，它较自动发现功能，更适用于跨区域的大范围的自动添加被监控主机时使用。

然而，Zabbix 系统服务器端，在自动添加被监控设备代理自动注册功能发现的主机时，只能根据被监控设备代理组件发来的信息，获取将要被自动添加主机的被监控设备代理端组件的信息。所以，系统只能自动地在被添加的主机上，添加上通过被监控设备代理采集监控数据的数据采集接口。而其他类型的监控数据采集接口，比如 SNMP 协议监控数据采集接口、JMX 协议监控数据采集接口和 IMPI 协议监控数据采集接口等，系统都没有办法自动地在被添加的主机上添加。因此，如果通过被监控设备代理自动注册功能自动添加进监控系统的主机上，存在有需要通过上述这些不能被自动添加的监控数据采集接口时，则使用这些接口的监控项目将无法自动添加成功。

通过上述的介绍不难理解，所谓被监控设备代理注册事件，就是由未在监控系统中配置的主机上所运行的被监控设备代理，向 Zabbix 服务器端发送自身“存活”的消息，从而在 Zabbix 系统中所产生的事件。

4.6.4 内部事件 (Internal events)

在 Zabbix 2.2 以后的版本中，增加了一种新类型的事件，即“内部事件 (Internal events)”。该类型事件会在以下情况下产生：

- 当有监控项目的状态从“正常”转变为“不支持”时。

- 当有监控项目的状态从“不支持”转变为“正常”时。
- 当有自动发现规则从“正常”状态转变为“不支持”状态时。
- 当有自动发现规则从“不支持”状态转变为“正常”状态时。
- 当有触发器的状态从“正常”转变为“未知”时。
- 当有触发器的状态从“未知”转变为“正常”时。

4.7 动作行为升级

通过前面章节的介绍，我们对动作行为的升级有了一个大致的了解。在这里我们对 Zabbix 系统这个功能的机制和作用做进一步的介绍和探讨。

我们知道，系统中每产生一个事件，都有可能触发某个动作（是否会触发动作，将依赖于系统是否配置了针对该事件响应的动作，以及动作的触发条件配置情况），而只要动作被触发，则就会相应地执行某些特定的行为，也就是动作具体要做的事务，比如给指定的用户或用户组中的用户发送报警信息，在指定的远程主机上执行指定的远程命令，将一台新发现的主机添加到系统，将主机添加到指定的主机组或将指定的模板关联到指定的主机，等等。所以，在 Zabbix 系统中，应该可以说，动作的行为是一个动作之所以需要被创建和配置的意义所在。离开了行为，则对应的动作就什么也干不了，因而对应的动作基本上没有存在的意义了。

而在 Zabbix 系统中，动作的行为可以是某个单独的行为，也可以是由一系列不同的行为组成的行为组。而不管构成某个动作行为的是单个行为，还是由一系列不同的行为组成的行为组，这些行为都是可以按照一定的时间间隔被循环执行的。当然，也可以只被执行一次，还可以在不同的时刻执行不同的具体行为。因此，在 Zabbix 系统中，动作行为的配置是非常灵活的。可以通过配置动作行为规则，来实现个性化的报警机制和故障处理机制。应该说，这也是 Zabbix 系统功能强大的一个体现吧。具体来说，通过配置动作中不同行为规则，可以使动作实现如下功能：

- 可以通过配置动作行为实现，当 Zabbix 系统检测到有被监控主机或被监控项目发生新的故障时，立即给指定的用户或用户组里用户发送报警信息。
- 报警信息可以按一定的时间间隔重复发送，直到故障被解除。
- 报警信息也可以被实现为延时发送。即当系统检测到某个故障时，可以延时一定的时间，如果延时的时间内该故障还没有恢复，则给指定的用户或用户组里的用户发送报警信息。这个功能，在针对某些类型的服务监控时比较有用。比如说在对系统负载进行监控与报警时，就可以使用延时发送报警信息的功能。因为，有时候，某些系统会在短时间内有大量的突发访问量，这个时候可能会引起系统负载短时间内超过阈值。而一旦突发的访问量降下来了，系统负载值又会很快降低到正常值范围内。在类似这种场景下，就可以使用这种延时报警功能。因为，在这种场景下，如果不使用延时报警功能，而是只要系统负载超过我们所设定的阈值就立即报警，那么，很有可能当有相关人员收到报警信息后，登录到被监控主机系统上进行检查或者尝试排除故障时，系统负载已经回落到正常值了。而且，一般的服务器都是可以允许系统负载在较短的时间内运行在较高位置上的。况且，只要系统负载值不是太高，很短的时间内系统负载超过理想值，并不会对用户的体验产生太大的影响。所以，在这种情况下，确实不需要立即处理这类故障。但是，如果系统负载值在较长时间内超过理想值，那么这就

有可能会导导致系统无法正常地对外提供服务。所以，在这种情况下，我们又要求监控系统能够将报警信息发送给相关人员。

- ❑ 可以实现报警信息在延时一定的时间后，发送给更“高级”用户。为什么要实现这个功能呢？或许在如下这个场景下，实现这个功能就非常有必要。比如说，我们将公司里的服务器，按照它们的用途分配给不同的系统管理员进行日常管理和维护。显然，这些服务器一旦出现故障，或者运行在这些服务器上的服务出现了故障，则监控系统当然应该将报警信息发送给管理该服务器的系统管理人员。但是，系统管理人员很可能在一定时间内，并不能完全排除系统故障，或者由于某些原因，系统管理人员并没有接收到相应的报警信息。那么，在间隔了一定的时间后，就要要求监控系统将相应的故障信息发送给系统管理员的主管或更高一级的领导，由主管或更高一级的领导来安排其他人员及时处理发生的故障。
- ❑ 可以延时一定的时间或立即执行远程命令。
- ❑ 可以实现当故障排除后，给某些用户发送故障恢复消息。

Zabbix 系统中的动作之所以可以实现上述这些功能，是因为 Zabbix 系统中的动作引入了动作升级的机制。每个动作在被触发后，都会按照一定的时间间隔进行升级，直到引发动作被触发的触发器的状态重新转变为“正常（OK）”时，动作行为的升级过程才会停止。细心的读者可能已经注意到了，我们这里所说的是，直到被触发的触发器状态重新转变为“正常（OK）”时，动作行为的升级过程才会停止。换句话说，虽然在 Zabbix 系统中，有多种类型的事件都可以触发相应类型的动作。但是，只有事件类型为触发器类的事件所触发的动作，其行为才会升级。而其他类型的事件，例如自动发现类事件及自动注册类事件，所触发的动作，其行为则不会升级。所以，这些类型事件所触发的动作，其行为只会被执行一次。

动作行为在升级过程中，每升级一个步骤都对应一个时间间隔。而这个时间间隔可以为动作设置的默认时间间隔，也可以是针对每一步行为独立设置的时间间隔。动作行为的最小升级时间间隔是 60 秒。在动作行为升级过程中，每一步所对应的行为都可以被设置成发送报警信息或执行一个远程命令。当行为的步骤号被设置为 1 时，则表示对应的行为需要立即执行而不延时，所以如果要想延时执行一个行为，则需要将其步骤号设置为大于 1 的数字。而如果将某个行为的截止步骤号设置为 0，则表示对应的行为需要重复执行下去，直到故障被排除。但是在不同的情况下，动作行为的升级有着不同的特性。表 4-3 列出了动作行为在不同的场景下，其所具有的升级特性。

表 4-3 不同场景下的动作行为升级特性列表

场景	动作行为
在主机因为故障在 Zabbix 系统发送完首次报警信息后进入维护状态时	被触发动作余下的行为仍然会被继续执行。将主机设置成维护模式，不能停止已经被触发动作行为的继续执行。但是，主机的维护模式可以影响动作是否会被触发。换句话说，如果在动作还没有被触发之前，主机就进入了维护状态，则这个时候，对应的动作可能将不再会被触发（是否会被触发需要依据动作中所配置的触发条件而定）
当动作条件中设置的时间区间的截止时间在该动作首次发送完报警信息之后	被触发动作余下的行为仍然会继续执行。在动作条件中的时间区间内不能停止已经被触发动作中行为的执行，但是它可以影响动作是否会被触发

续表

场景	动作行为
当维护期间产生故障，且维护时间结束后，故障仍然没有被恢复时	则动作行为的升级将从维护状态结束时刻开始执行
当不带数据采集的维护过程中出现了故障，且维护时间结束后故障仍然没有排除时 ¹	在与动作相关的触发器的状态被确认之前，动作行为的所有升级步骤均会继续执行
不同的行为级别是逐级升级的，即低一级行为被执行成功后才执行高一级行为	高一级行为的执行会替代前面低一级行为的执行，即高一级的行为是在前面低一级的行为执行完成后才会被执行，且高一级行为会替代低一级行为
行为升级正在进行的过程中（例如正在发送报警信息时）动作被禁止了	正在处理中的信息将会继续被执行，然后接下来，系统会发送一条或数条行为被升级的信息。之后系统会紧接着发送一条内容中包含“注意，行为升级已经取消：动作<动作名>被禁止”的信息。通过这种方法可以让用户知道动作行为的升级已经被取消了，对应动作中的行为不会再执行升级步骤了

4.8 数据映射

在我们配置某些监控项目时，可能会遇到这样的情况，该监控项目的监控数据可能会是多种状态值，而每种状态都是由 0、1、2……等无符号的整数所表示。例如，当监控戴尔的某个型号的网络存储磁盘状态时，戴尔存储系统就可能针对每个磁盘插槽的状态，给监控项目返回如表 4-4 所示的多种状态值。

表 4-4 戴尔网络存储磁盘状态值列表

返回值	所代表的状态
1	代表磁盘工作正常
2	表示磁盘处于空闲状态，即没有被划入到任何卷组中
3	代表磁盘工作故障
4	表示对应的插槽没有插入磁盘
6	磁盘空间已满
7	磁盘之前发生过故障，现在已经恢复
8	插槽中被插入了不被支持的磁盘
9	磁盘发生非严重故障，比如读写出错等
10	磁盘更换中

¹ 所谓不带数据采集的维护，是 Zabbix 系统中可以配置的两种维护模式中的一种。处在这种维护模式下的主机，系统将不会再对其上所配置的监控项目采集监控数据。

由表 4-4 可以看出，如果监控戴尔网络存储系统的磁盘状态，那么监控系统在针对它监控数据时，就可能采集到表 4-4 所列的 10 个整数中的一个。而如果监控系统直接将这些采集到的，代表磁盘状态的数字显示在系统相关页面上，我相信，绝大多数人都无法时时记得住这些数字所代表的磁盘状态。而如果通过查阅戴尔所提供的技术资料，在 Zabbix 系统中配置好每个表示磁盘状态的数字与对应状态文字描述之间的映射关系，然后 Zabbix 系统在显示这些状态时，直接显示每种状态所对应的文字描述，以替代表示状态的数字。这样，就可以很清楚、方便地查看所监控的磁盘状态，且不需要每次都查阅戴尔提供的技术文档或者记住这些十分难记的状态对应关系了。在 Zabbix 系统中，像这种表示状态的数字与表示状态的文字描述之间的对应关系就叫作数据映射。

当在 Zabbix 系统中，为某个监控项目配置了数据映射后，Zabbix 系统将不仅在显示该监控项目的监控数据时使用该数据映射，而且在发给用户的报警信息中也使用映射后的数据。但是，在 Zabbix 系统数据库中，仍然存储的是系统所采集到的原始监控数据，而不是映射后的数据。同时需要注意的是，只有监控项目的信息类型是数值型，包括无符号整型和浮点型时才可以使用数据映射。

数据映射的配置很简单，依次选择“高级配置”→“常规”菜单项，然后在“配置图形界面”页面上，从右边上部的下拉菜单里选择“数据映射”菜单项，系统就会打开“配置数据映射”列表页面。该页面上，显示了系统里已经配置的数据映射信息列表，如图 4-1 所示。

配置数据映射	
数据映射	
名称	数据映射
APC Battery Replacement Status	1 → unknown
	2 → notInstalled
	3 → ok
	4 → failed
	5 → highTemperature
	6 → replaceImmediately
	7 → lowCapacity
APC Battery Status	1 → unknown
	2 → batteryNormal
	3 → batteryLow
Dell Open Manage System Status	1 → Other
	2 → Unknown
	3 → OK
	4 → NonCritical
	5 → Critical
	6 → NonRecoverable

图 4-1 数据映射列表截图

在图 4-1 所示的页面上，单击页面右上部的“新建数据映射”按钮，系统将打开“配置数据映射”页面，如图 4-2 所示。在“配置数据映射”页面上，就可以创建新的数据映射信息。“配置数据映射”页面上的表单项比较简单，该页面上各表单项的含义为：

- 名称 (Name)。数据映射组的名称，该名称在系统中必须是唯一的，不能有重复。
- 值 (Value)。为映射组中单项的数值，即监控系统可能采集到的监控数据的原始值。
- 映射为 (Mapped to)。即为对应项映射后的字符或字符串内容。

很显然，在完成数据映射的配置后，Zabbix 系统并不知道哪些被监控项目需要使用数据映射。所以，需要在配置监控目时，指定该监控项目是否需要使用数据映射，以及该监控项目需要使用哪个数据映射规则。将数据映射规则应用到某个监控项目上的方法也很简单，只需在配

置监控项目时，在“配置项目”表单页上的“显示值”表单项中选择需要的数据映射规则即可，如 **Dell存储磁盘状态** 显示数据映射。

名称 **Dell存储磁盘状态**

映射	值	映射为	
	<input type="text" value="1"/>	→ 正常	移除
	<input type="text" value="2"/>	→ 空闲磁盘	移除
	<input type="text" value="3"/>	→ 故障	移除
	<input type="text" value="4"/>	→ 没有插硬盘	移除
	<input type="text" value="6"/>	→ 太小	移除
	<input type="text" value="7"/>	→ 历史故障	移除
	<input type="text" value="8"/>	→ 不支持的硬盘	移除
	<input type="text" value="9"/>	→ 硬盘问题	移除
	<input type="text" value="10"/>	→ 硬盘替换中	移除
添加			

图 4-2 配置数据映射表单截图

4.9 宏（Macro）及宏的替换顺序

对于学习过任何一门高级编程语言的读者来说，宏（Macro）的概念并不陌生。在许多高级编程语言中，比如 C 语言、PHP 语言等，都会看到宏的身影。如果对宏没有任何概念，那也没有关系，这里会简单介绍什么是宏。如果简单来理解宏，可以将宏视为一种特殊的变量，该变量一旦定义好后，在程序中就不可以再去修改它所指代的值，而只能去使用它，且在程序运行的整个过程中，所有引用宏的地方，系统都会使用该宏所指代的值来替换该宏。

举个例子来说，如果在程序或某个应用系统中定义了宏 {SMACRO}=10，那么在这个应用系统或程序运行时，系统或程序中所有包括字符串“{SMACRO}”的地方，都会被系统或程序自动替换成数字“10”，这就是宏的作用。当然，这里只是对宏做了简单的解释，如果在程序设计语境里，宏的作用和含义要远远比这复杂得多，读者如果需要对宏做进一步的了解，可以参考相关的编程书籍。在使用 Zabbix 系统时，只需将宏理解为一种特殊的变量即可。

在 Zabbix 系统中，用户可以定义 3 种级别的宏。它们是全局宏、模板（什么是模板，我们在后续章节中有详细介绍）级别的宏和主机级别的宏。在 Zabbix 系统中，定义宏的语法为 {SMACRO} 形式。其中，MACRO 为宏的名称，用户可以根据自己的需要和喜好定义宏的名称。但是，宏的名称只能是由 A-Z、0-9、_、. 等字符组成的字符串。这里需要注意的是，组成宏名称的英文字母必须是大写的，不可以是小写的英文字母，而且宏名称左右两边的大括号（{}）和美元符号（\$）也都是宏的组成部分，不管是在定义宏的时候，还是引用宏的时候，这些大括号和美元符号都必须要有，这一点跟某些编程语言中定义和使用的宏变量略有不同。

在 Zabbix 系统中，已经定义好了的宏可以在下列这些地方使用：

- 在监控项目名称中可以使用宏。当在监控项目名称中使用宏时，则系统在显示该监控项目的名称时，会用宏所指代的值替换宏。
- 监控项目的关键字参数中可以使用宏。

- ❑ 触发器名称中可以使用宏。
- ❑ 触发器表达式中的参数和常量可以使用宏。
- ❑ 其他可以使用宏的地方。具体还有哪些其他地方可以使用宏，请参考本书的附录 D。

前面介绍过，在 Zabbix 系统中，用户可以定义 3 种级别的宏。它们是全局宏、模板级别的宏和主机级别的宏。从这 3 种级别的宏称谓上来看，不难看出它们分别的作用范围。全局宏，当然是 Zabbix 系统中所定义的所有主机、模板和触发器等都可以使用它；而模板级别的宏，则只有关联了对应模板的主机或模板（模板之间也是可以关联的）才可以使用它；相应的，主机级别的宏则只有本主机上的监控项目和触发器才能使用它。如果要定义全局宏，则只需依次选择“高级配置”→“常规”菜单项，并在“配置图形界面”页面的右上部的下拉菜单里选择“宏变量”菜单项，系统就会打开“配置宏”的表单页面，如图 4-3 所示。

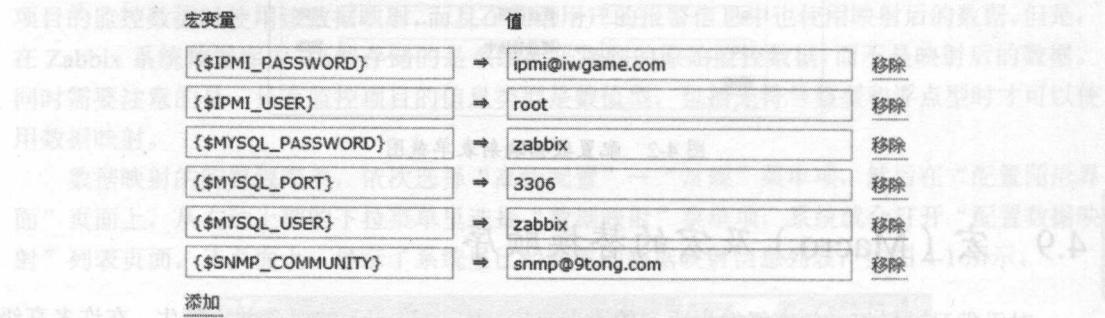


图 4-3 配置宏变量表单页面截图

“配置宏”页面上的表单很简单，单击该表单下部的“添加”超链接，就可以在页面上增加新的配置全局宏变量的表单项；而单击宏变量表单项后面的“移除”超链接，则可以删除对应的宏变量。而“宏变量”和“值”表单项则用于输入所定义的宏变量和它所对应的值。

在配置模板和配置主机的页面上都有一个叫作“宏变量”的选项卡，单击这个选项卡，系统同样也会显示一个如图 4-3 所示的宏变量配置表单。通过这个表单，就可以配置模板级别和主机级别的宏变量。它们的配置方法与全局宏变量的配置方法是类似的，操作也十分简单，在此就不一一介绍了。

或许你会问，既然可以在 Zabbix 系统中定义全局的、模板级别的和主机级别的 3 个级别的宏。那么，可以分别在这 3 个级别上，定义同一个宏（宏名称相同），但是宏所指代的值不一样吗？如果可以，那么 Zabbix 系统将取哪一个级别上宏的值作为最终的值来替换对应的宏呢？是的，我们完全可以在全局、模板和主机这 3 个级别上都定义同一个宏，且宏的值互不相同。至于，Zabbix 系统最终会取哪一级别宏的值，作为最终的值来替换对应的宏，其实质就是 Zabbix 系统中宏的替换顺序问题。在 Zabbix 系统中，宏的替换是按照下面所述的顺序进行的：

❶ 当某个监控项目或触发器中有宏变量需要被替换时，系统将首先查看该监控项目或触发器所在主机上宏的配置情况，如果在本主机上能找到对应的宏，则系统将会以该宏的值来替换对应的宏，并且停止进行下一步的匹配。如果在本主机上没有找到对应的宏，则 Zabbix 系统会接着执行下面的过程。

❷ 如果在第一步的匹配过程中，系统没有找到对应的宏变量，则系统就会在第一级模板（也就是直接关联到对应主机上的模板）上查找指定的宏变量。同样，如果系统在这一级的模板中找到了指定的宏变量，则系统将会以该宏变量的值来替换对应的宏，并且停止接下来的匹

配过程。否则，Zabbix 系统会接着执行下面的匹配过程。

③ Zabbix 系统会继续匹配关联到该主机的第二级模板（也就是通过其他模板间接关联到某台主机的模板）。而第二级模板的匹配是以关联到对应主机上的所有第 2 级模板的模板 ID 值从小到大的顺序进行的。一旦系统找到指定的宏变量，则会以该宏变量的值来替换对应的宏变量，并停止后续的匹配步骤。

④ 如果系统在关联到对应主机的第 2 级模板中时，都没有找到指定的宏变量，则系统会继续匹配关联到该主机上的所有第 3 级模板中所配置的宏变量。与匹配第 2 级模板中的宏变量一样，匹配第 3 级模板中的宏变量时，匹配也是以关联到对应主机上的所有第 3 级模板的模板 ID 从小到大的顺序进行的。同样，如果系统找到了指定的宏变量，则会以所找到的宏变量的值来替换对应的宏变量，并停止后续的匹配步骤。如果在所有第 3 级模板中仍然没有找到指定的宏变量，则系统会重复这样的匹配操作，以匹配第 4 级、第 5 级……第 N 级模板，直到将所有关联到指定主机上的模板中所配置的宏变量都匹配一遍。

⑤ 如果在上述所有关联到指定主机上的模板中都没有找到指定的宏变量，则系统最后会在全局宏变量中进行查找。同样，如果系统在全局宏变量中找到了指定的宏变量，则会以该宏变量的值来替换所对应的宏变量。而如果在上述所有的匹配过程中，系统都没有找到所指定的宏变量，则系统会放弃替换，并在相关的显示页面或报警信息中保留宏变量的名称形式。

从上面的描述中不难看出，在 Zabbix 系统中的全局、模板和主机 3 个级别的宏变量中，主机级别的宏变量被替换的优先级最高，其次是模板级的宏变量，全局级的宏变量被替换的优先级最低。基于这个机制，我们可以在配置模板中的触发器时，对于触发器的触发阈值使用宏变量，然后在关联它的不同主机上配置相同名称的宏变量，但是宏变量的值可以因被监控主机的不同而设置成不同的值。这样做，可以实现，对于同一类触发器，在不同的主机上有不同的触发阈值，从而实现不同主机上同类监控项目有不同报警阈值的功能。

提示：如果配置在模板上的监控项目或触发器中使用了宏变量，则即使已经在系统中定义了相应全局级宏变量，但仍然建议在该模板上定义一个同名称的模板级宏变量。这样做的好处是，如果将这个模板导出成 XML 文件，并在另外一个 Zabbix 系统中导入，则它不会因为在新导入的系统中没有定义相应全局宏变量，而出现意想不到的结果。

不管是全局级别的宏变量还是主机级别的宏变量，它们都各有自己的优势。例如，定义和使用主机级的宏变量，不但可以实现前面所说的，相同种类的监控项目针对不同的主机具有不同的报警阈值。而且，还可以实现，相同种类的监控项目，在不同的主机上使用与特定主机相关联的特定参数的功能。例如，下面的监控项目实现的是对主机上 SSH 服务端口的监控。通过这个例子就可看出，通过主机级别的宏变量，可以实现相同种类的监控项目，在不同的主机上使用与特定主机相关联的特定参数的功能。

例 1: net.tcp.service[ssh,{SSH_PORT}]

不难看出，在这个例子中使用了宏变量 {SSH_PORT}。这样，如果将这个监控项目添加到某个模板上，并将该模板关联到不同的主机上，同时在关联了该模板的对应主机上配置好宏变量 {SSH_PORT}，并使该宏变量所指代的值等于对应主机上所运行的 SSH 服务的端口号。则通过这种方法，就实现了，同一个模板关联到不同的主机，并对不同的 SSH 服务端口进行监控。

而全局宏变量的优势就在于，它维护和修改起来非常方便。一旦需要修改某个全局宏变量

所指代的值，不需要一台一台主机地修改，只需修改这个全局宏变量所指代的值就可以了。

接下来，再举几个使用宏变量的例子。

例 2: `{system.cpu.load[,avg1].last(0)}>{$MAX_CPULOAD}`

功能：最新所采集到的系统一分钟平均负载的值大于宏变量 `{$MAX_CPULOAD}` 所指定的值时，对应触发器就被触发。

例 3: `{system.cpu.load[,avg1].min({$CPULOAD_PERIOD})}>{$MAX_CPULOAD}`

功能：在宏变量 `{$CPULOAD_PERIOD}` 所指定的时间或所采集的监控数据个数中，若系统一分钟负载的最小值，大于宏变量 `{$MAX_CPULOAD}` 所指代的值时，则该触发器就会被触发。

提示：在触发器函数 `min`、`max` 等中，参数既可以是表示时间的数字（直接用数字表示），也可以是系统所采集到的监控数据个数，用“#数字”形式表示（即井号#加数字的形式，如#3、#5，等等）。如果在这些函数中使用宏变量，且函数中参数取所采集到的监控数据个数而不是时间时，则这些宏变量的值也应被定义成“#数字”形式。例如，上例中的宏变量 `{$CPULOAD_PERIOD}`，如果该触发器中的函数 `min` 的参数打算使用系统所采集到的监控数据的个数，则宏变量 `{$CPULOAD_PERIOD}` 所指代的值的前面也应加上“#”符号，如 `{$CPULOAD_PERIOD}→#3` 的形式。

警告：如果要在触发器表达式中使用宏变量，则该宏变量所指代的值只能是常量或参数，而不能是一个主机的名称、监控项目的关键字、函数、操作符或其他触发器表达式等内容。

4.10 Zabbix 系统报警流程分析

笔者认为，如果要掌握一款监控软件，那么它的报警流程和报警机制则是我们必须学习 and 研究的重点内容之一。应该说 Zabbix 系统的报警流程还是比较复杂的，因为，首先它有多种类型的事件驱动。其次，它有独特的行为升级机制，特别是对于同一个事件，在不同的时间阶段，可以将不同的信息发送给不同的用户的机制。应该说，这个机制在许多开源监控系统中都是没有的（至少笔者所了解的开源监控系统中除了 Zabbix 以外，其他系统都没有这个机制）。再次，在 Zabbix 系统的报警流程中，用户可以设置灵活和复杂的触发条件，且这些条件还可以支持逻辑运算，而这也是许多监控软件所不具备的。最后，在 Zabbix 系统的报警流程中增加了用户权限控制功能。笔者在编写本书时，曾经花了不少时间查找 Zabbix 系统的官方资料，希望能从 Zabbix 系统的官方资料中找到 Zabbix 系统的报警流程图。但是，遗憾的是，笔者没有找到官方给出的 Zabbix 系统报警流程图。故此，笔者根据自己对 Zabbix 的掌握和研究，结合相关资料，绘制出了 Zabbix 系统报警流程图，如图 4-4 所示。

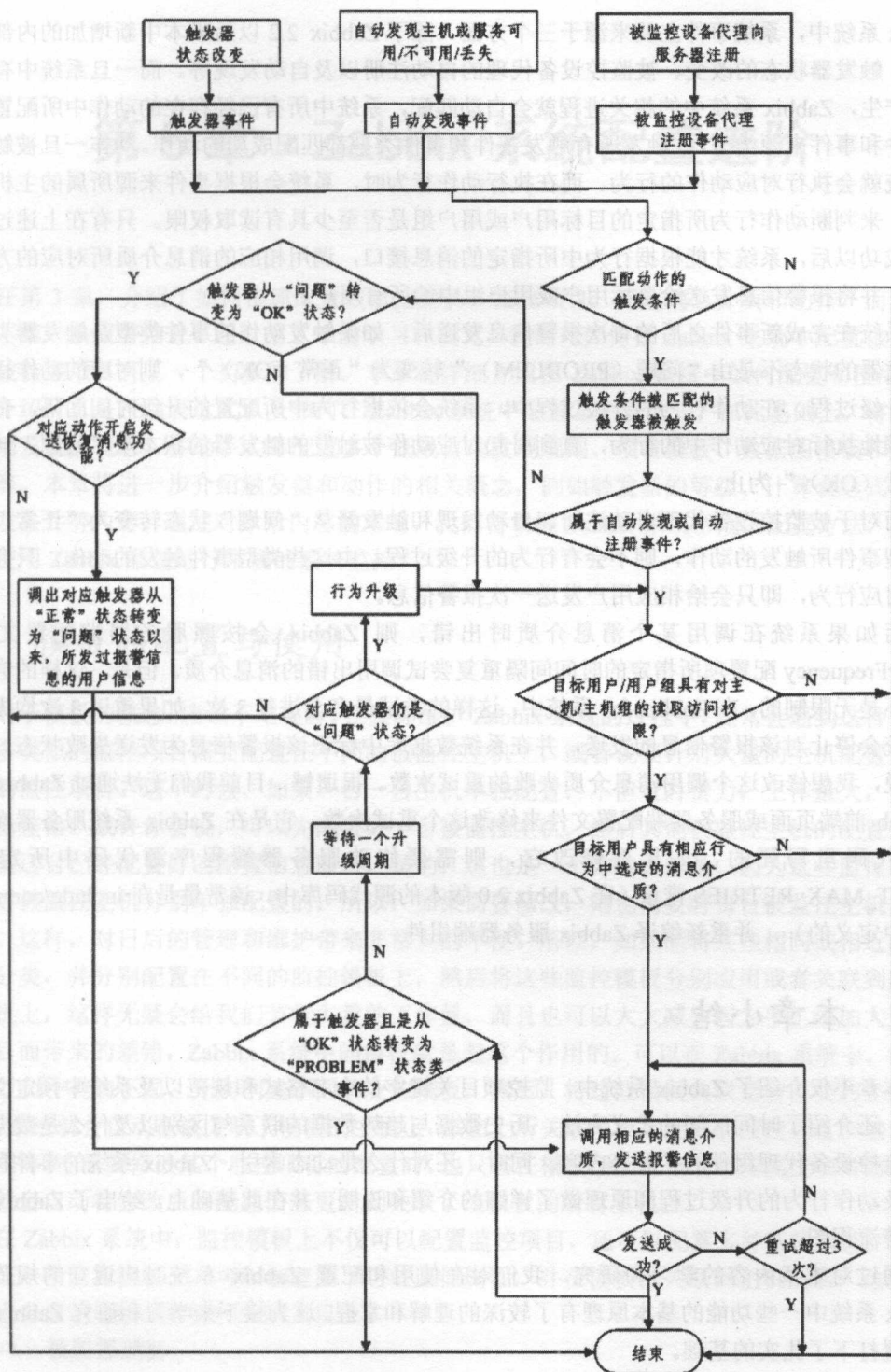


图 4-4 Zabbix 系统报警流程图

如我们在前文中所述的，Zabbix 系统中的报警是通过系统中的事件来驱动动作，而后由动作来执行相应的行为，最后由行为通过指定的消息介质接口将报警信息发送给用户的。而在

Zabbix 系统中，系统事件主要来源于三个方面（除了 Zabbix 2.2 以后版本中新增加的内部事件外）：触发器状态的改变、被监控设备代理的自动注册以及自动发现等。而一旦系统中有新的事件产生，Zabbix 系统中的相关进程就会自动匹配，系统中所有已经存在的动作中所配置的触发条件和事件来源类型，并触发所有触发条件和事件类型都匹配成功的动作。动作一旦被触发，则系统就会执行对应动作的行为。而在执行动作行为时，系统会根据事件来源所属的主机或主机组，来判断动作行为所指定的目标用户或用户组是否至少具有读取权限。只有在上述过程都匹配成功以后，系统才能根据行为中所指定的消息接口，调用相应的消息介质所对应的方法和程序，并将报警信息发送给目标用户或用户组中的所有用户。

系统在完成新事件之后的首次报警信息发送后，如果触发动作的事件类型是触发器类的，且触发器的状态不是由“问题（PROBLEM）”转变为“正常（OK）”，则对应的动作行为将进入升级过程。在动作行为的升级过程中，系统会依据行为中所配置的升级时间间隔，有计划有步骤地执行对应动作中的行为，直到引起对应动作被触发的触发器的状态从“问题”转变为“正常（OK）”为止。

而对于被监控设备代理自动注册、自动发现和触发器从“问题”状态转变为“正常”状态等类型事件所触发的动作，则不会有行为的升级过程。由这些类型事件触发的动作，只会执行一次对应行为，即只会给相应用户发送一次报警信息。

而如果系统在调用某个消息介质时出错，则 Zabbix 会按照服务器端配置文件中 `SenderFrequency` 配置项所指定的时间间隔重复尝试调用出错的消息介质。但是，这样的尝试次数并不是无限制的，在 Zabbix 2.0 系统中，这样的尝试最多会进行 3 次。如果重试 3 次均失败，则系统会停止对该报警信息的发送，并在系统数据库中标识该报警信息为发送失败状态。或许你会说，我想修改这个调用消息介质失败的重试次数。很遗憾，目前我们无法通过 Zabbix 系统的 Web 前端页面或服务器端配置文件来修改这个重试次数。它是在 Zabbix 系统服务器端程序的源代码里写死的，如果要修改它，则需要修改服务器端程序源代码中所定义的 `ALERT_MAX_RETRIES` 常量（在 Zabbix 2.0 版本的源代码库中，该常量是在 `include/common.h` 文件中定义的），并重新编译 Zabbix 服务器端组件。

4.11 本章小结

本章不仅介绍了 Zabbix 系统中，监控项目关键字的命名格式和规范以及系统中预定义的关键字，还介绍了时间区间的定义方法、历史数据与趋势数据的联系与区别以及什么是数据映射和被监控设备代理组件的扩展等内容。同时，还对什么是动态索引、Zabbix 系统的事件和事件源以及动作行为的升级过程和原理做了详细的介绍和说明，并在此基础上，绘出了 Zabbix 系统中报警流程图。

通过对本章内容的学习和研究，我们对在使用和配置 Zabbix 系统时应遵守的规范以及 Zabbix 系统中一些功能的基本原理有了较深的理解和掌握。这为接下来学习和研究 Zabbix 系统的配置打下了扎实的基础。

第5章 Zabbix 系统配置进阶

在第3章，介绍了如何快速配置被监控主机、监控项目，配置消息介质和动作，而且还接收到了第一条报警信息。通过第3章的学习和研究，我们对如何在 Zabbix 系统中配置被监控主机和被监控项目有了一个初步的了解。本章将详细介绍在 Zabbix 系统中如何配置和使用模板，并将介绍什么是正则表达式以及如何在 Zabbix 系统中配置正则表达式，在此基础上，将介绍低级自动发现的概念，并动手实际配置多个低级自动发现规则、项目模板、数据图模板和触发器模板等。本章将进一步介绍触发器和动作的相关概念，例如触发器的等级、计算表达式、动作的触发条件等内容。通过对本章内容的学习，我们将掌握 Zabbix 系统常规的配置方法，并能够熟练使用 Zabbix 系统进行常规的系统监控。

5.1 模板的配置与使用

对于模板的概念，应该不难理解。在实际维护 Zabbix 系统的过程中，经常会遇到这种情况。有很多类似的监控项目需要配置在不同的被监控主机上，或者说要针对大量的主机配置相同或类似的监控项目。这个时候，如果一台一台主机单独配置，不但费时费力，工作量大，而且还容易出差错。或许你会说，可以先配置好一台被监控主机，然后其他被监控主机的配置信息只需复制这台已经配置好的配置信息即可。是的，这也是一个方法。但是，因为这些监控项目都是针对被监控主机分别单独配置的，所以，如果需要修改，则也需要时每台被监控主机分别做修改。这样，对日后的管理和维护带来非常大的不便。所以，如果能将某些相同或相近的监控项目分类，并分别配置在不同的监控模板上，然后将这些监控模板分别应用或者关联到所需要的主机上，这样无疑会给我们节省大量的工作量，而且也可以大大减少因为手工添加大量的监控项目而带来的差错，Zabbix 系统中的模板就是起这个作用的。可以在 Zabbix 系统中，根据业务类型和种类的不同，分别创建不同的业务模板。然后，将这些模板关联到所需要的主机上。这样，系统就会自动在被关联了模板的主机上，添加上所关联模板上配置的监控项目和触发器、数据图等。当需要修改某个监控项目的配置时，也只需在模板层面上修改对应监控项目的配置，系统就会自动将修改后的配置信息更新到关联了该模板的所有主机上。

在 Zabbix 系统中，监控模板上不仅可以配置监控项目，还可以配置多种监控属性。具体来说，在一个监控模板上，可以创建如下这些监控属性，并在模板被关联到主机上后，这些监控属性也会自动地被关联到相应的主机上：

- 被监控项目
- 触发器
- 数据图
- 分类
- 图表（只有 Zabbix 2.0 及以上的版本中，模板上才可以创建图表）

□ 自动发现规则（只有 Zabbix 2.0 及以上的版本中，模板上才可以创建自动发现规则）

或许你会说，如果在模板上配置触发器，那么关联了这个模板的所有主机上的该触发器不是都使用相同的阈值了？但是，如果想使从模板上关联过来的触发器，在不同的主机上具有不同的触发阈值，该怎么办呢？是的，如果在配置模板时，将触发器上的触发阈值写死了，那么关联了该模板的所有主机上的对应触发器都会使用相同的触发阈值。但是，在 Zabbix 系统中，有一个变通的方法，那就是使用我们在前面章节中介绍的宏变量。只需在配置模板上的触发器时，指定它的触发阈值为一个宏变量，然后针对关联了该模板的所有主机，分别定义主机级别的宏变量。这样，根据我们在前一章中介绍的宏变量的替换顺序，我们不难做到，让同一类触发器，在不同的主机上具有不同的触发阈值。如果对应触发器的触发所产生的事件，会驱动一个报警动作的话，那么就可以实现同一类触发器，对于不同的主机具有不同的报警阈值了。当然，也可以定义对应的全局宏变量和模板级别的宏变量。这样，如果关联了对应模板的主机上没有定义主机级别的宏变量，则系统就会使用模板级别的宏变量或全局级别的宏变量来替换触发器阈值所引用的宏变量，从而可以实现所谓报警默认阈值的功能。

在 Zabbix 系统中，一个模板不但可以被关联到一个或多个主机上，而且模板之间也是可以关联的。也就是说，一个模板可以关联到系统中存在的另一个模板。当关联了另一个模板的模板被关联到一个主机上时，系统会将它所关联模板上的配置信息（例如监控项目、触发器、数据图等）也一同关联到对应主机上。但是，模板之间的关联不能循环或重复关联。循环关联可能读者还比较容易理解，但是什么叫重复关联或者说什么样的关联就是重复关联呢？读者可能一时不太容易理解。这里举个例子来加以说明，或许读者就很容易明白了。假设，在我们系统中有 3 个模板，分别是模板 A、模板 B 和模板 C。如果模板 B 被关联到了模板 A 上，此时，如果模板 B 再去关联模板 A，则叫作模板的循环关联。在 Zabbix 系统中，模板之间是不允许这样循环关联的。而假如模板 B 关联了模板 A，且模板 C 又关联了模板 B，此时如果再试图将模板 A 关联到模板 C，这就叫作模板的重复关联，这样的关联尝试是不会成功的。因为模板 A 已经通过模板 B 关联到模板 C 上了，如果这个时候再将模板 A 关联到模板 C，那实际上模板 C 对模板 A 做了两次关联。实际上，不但模板之间的关联不能重复和循环，而且当一个主机或模板上关联了多个模板时，这些被关联的模板之间不能有相同的监控属性。包括不能有相同的监控项目（即使监控项目的名称不同，但是监控项目的关键字相同，也会被系统视为是同一个监控项目）、相同名称的分类、相同名称的图表、数据图等，否则在添加关联模板时系统就会报错。所以，我们在创建模板之前，做好模板的规划与设计就显得非常重要，否则随着时间推移，我们在系统中创建的模板就会越来越多，且系统中模板之间的关联关系也会越来越复杂，最后会导致系统中模板管理混乱。

在上面的叙述中，当描述模板与模板、主机与模板之间的关系时，我们使用的词是“关联”，而不是添加或者其他什么词语。这是因为，在 Zabbix 系统中，当一个主机关联了某个模板时，并不是将这个模板上的所有配置都复制一份到对应主机上，而只是建立了模板到主机的映射关系。所以，当要修改某台主机上来自于某个模板的监控项目或数据图等配置时，不能直接从该台主机上对相应的配置进行修改，而是需要修改对应模板上的对应监控属性。同时，如果对模板上的某个监控属性进行了修改和调整，则关联了这个模板的所有主机和模板上的对应监控属性都会自动被修改，包括那些通过多次模板关联过来的监控属性。

5.1.1 查看模板

如果想查看系统中已经配置了哪些主机模板，则只需在 Zabbix 系统的 Web 前端页面上依次选择“系统配置”→“模板”菜单，系统将会打开“配置模板”页面。在“配置模板”页面上，系统将显示当前系统中已经存在的模板信息，如图 5-1 所示。

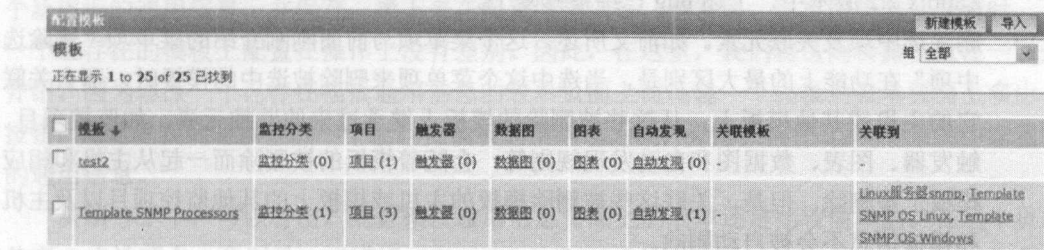


图 5-1 “配置模板”页面截图

在图 5-1 所示的“配置模板”页面上，系统显示了每个模板的名称、项目分类数、所拥有的监控项目数、触发器、数据图数、图表数、自动发现规则数、对应模板所关联的其他模板列表以及关联了对应模板的主机和模板的列表等信息。单击上述内容所对应字段列上的超链接，则系统就会打开对应模板的对应属性的配置或列表页面。例如，如果单击图 5-1 所示页面上“SNMP_Interfaces”模板行上的“项目”超链接，则系统会打开“SNMP_Interfaces”模板上所配置的监控项目的信息列表。而当单击图 5-1 所示页面上，“模板”标题上的超链接时，可以切换该页面上列表的显示顺序。即，单击一次该标题上的超链接，系统将以模板名称的从小到大顺序显示查询到的模板列表信息；再单击一次，则系统将会以模板名称的从大到小的顺序显示查询到的模板列表信息，这一点与我们在前面介绍的浏览其他对象列表是类似的。这个页面的右上部有一个称为“组”的下拉菜单，通过这个下拉菜单，可以选择只列出属于某个组的模板列表信息。

或许大家早就注意到了，在如图 5-1 所示的“配置模板”页面上显示的每个模板名称前面都有一个复选框。勾选模板名称前面的复选框，可以通过该页面左下角的下拉菜单对所选定的模板进行相关操作。这个下拉菜单共有 3 个菜单项：导出选中项、删除选中项和删除选中项及关联元素。这 3 个选项的功能分别是：

- ❑ 导出选中项。将所选中的模块从系统中导出，并保存为 XML 格式的文件，该文件可以被导入到新的 Zabbix 系统中。当一次选中多个模板时，则被选中的多个模板的信息将会被导出到一个 XML 文件中。
- ❑ 删除选中项。从 Zabbix 系统中删除所选中的模板。如果这些被选中的模板被关联到了主机或其他模板上，则删除这些模板时，关联它们的主机或模板，从它们这里继承的监控项目、触发器、数据图、图表、自动发现规则等监控元素不会随着这些模板的删除而从相应主机或模板上被删除。当这些模板被删除时，则关联这些被删除模板的主机或其他模板上，会自动添加上从这些被删除模板上继承过来的监控元素，且这些元素与相应的模板在删除之前有所不同。不同之处表现在，这些监控元素不再与任何模板有关联，而与在相应的主机或模板上单独添加的监控元素一样。在选中这个菜单项来删除系统中模板时，不会删除关联它的主机或模板上相应的监控项目。这是 Zabbix 系统的官方手册中提供的说法，且这也是这个菜单项与下面将要介绍的菜单项“删除选中项及关联元素”在功能上的最大不同之处。但是，经笔者测试，在 Zabbix 2.0 版

本的 Web 前端组件中，这个菜单项的功能是有 Bug 的。实际上，在我们选择这个菜单项来删除模板时，关联它的主机和模板上相应的监控元素也会被自动删除。因此，实际上，在 Zabbix 2.0 版本的系统中，这个菜单项的功能与我们将要介绍的菜单项“删除选中项及关联元素”的功能是没有差别的。不过，幸运的是，经过笔者的测试，在 Zabbix 2.2 版本中，上述 bug 已经被修复了。

❑ **删除选中项及关联元素。**如前文所述，这个菜单项与前面刚刚介绍的菜单项“删除选中项”在功能上的最大区别是，当选中这个菜单项来删除被选中的模板时，那么关联它的主机或其他模板上，从这些被删除的模板上继承过来的监控元素，如监控项目、触发器、图表、数据图和自动发现规则等，会随着模板的被删除而一起从主机和相应模板上被删除。但是，关联这些被删除模板的主机或模板上的其他监控项目以及主机和模板信息不会被自动删除。

在“配置模板”页面的右上角还有两个按钮“新建模板”和“导入”。这两个按钮的功能，将在后续章节陆续介绍。

5.1.2 配置模板



当在图 5-1 所示的“配置模板”页面上单击列表中模板名称上的超链接，或者单击该页面右上部的“新建模板”按钮时，系统会打开如图 5-2 所示的配置模板表单页面。

图 5-2 模板通用信息配置表单页面截图

在前面章节中介绍过，在主机模板上，可以创建监控项目、触发器、图表、数据图以及自动发现规则等监控元素。但是，如果要新建一个模板，那么在具体配置这些监控元素之前，需要先配置模板的通用信息，如模板名称、所属组等。图 5-2 所示的表单即为配置模板通用信息的表单。实际上，不管是修改一个已存在模板的通用信息，还是创建一个新的模板，配置模

板通用信息表单页面上的表单项相差不大。只是，在修改模板通用信息表单页面上，多出“复制”、“完全复制”、“删除”、“Delete AND clear”几个按钮，其他表单项的内容完全是一样的。且不管是修改一个已存在模板的通用信息，还是创建一个新的模板，其操作对用户来说实际上没有什么本质上的不同。而且，当需要创建一个新的模板时，也是先在系统中配置好这个新模板的通用信息，并保存。接下来在配置监控项目、触发器、数据图等监控元素时，与在一个已存在的模板上配置在操作上没有差别。因此，在这里，我们将这两种操作放在一起进行介绍。因为修改一个已存在模板通用信息的表单页面上比新建一个模板的表单页面上多出几个按钮，在这里我们将以修改一个已存在模板通用信息为例，介绍如何在 Zabbix 系统中配置模板的通用信息。

从图 5-2 中，可以看出，配置模板通用信息页面上的表单还是比较简单的。在这个页面上共有 3 个选项卡，分别是：“模板 (Template)”、“关联模板 (Linked templates)”和“宏变量 (Macros)”选项卡。

其中，“模板”选项卡是用来配置对应模板通用信息的。表单项“模板名称 (Template name)”里所要填写的是模板在系统中的名称，该名称在系统中不能有重复。“显示名称 (Visible name)”这个表单项不是必填项，但如果填写了该表单项，则在模板列表页面和拓扑图页面上，系统将以该表单项中所输入的内容来显示对应模板。同样，模板显示名称在系统中也必须是唯一的。而“组”表单区域则是由左右两个多选框组成，左边的多选框里显示的是当前模板所属于的主机或模板组，而右边的多选框则显示的是系统中已经定义的，且尚未被选定为当前模板所属组的主机组列表。在右边多选框中选中一项或多项，单击  按钮，可以将所选定的主机组设定为当前模板所属于的组。类似的，当在左边多选框中选中一项或多项，然后单击  按钮，则表示将当前模板从对应组中脱离。如果在“新建组 (New group)”表单项中输入了内容，则在修改模板配置信息的同时，系统会自动添加上以该表单项中输入的内容为名称的新的主机组，并将当前所配置的模板添加到新创建的主机组中。“主机/模板”表单域则是用于配置哪些主机或模板关联到当前所配置的模板。当然，在配置模板时，也可以不指定关联当前模板的主机或其他模板，而在有需要的时候，才通过配置主机页面来关联所需要的模板。

如前文所述，在配置模板表单页面上有多个按钮。其中“保存”和“取消”按钮，相信读者都会知道它们是什么意思，在此不作过多的讨论。我们在这里需要详细介绍一下，配置模板表单页面上的其他几个按钮的作用以及它们的差别。

- **复制 (Clone)**：单击这个按钮，系统会以当前模板为基础创建一个新的模板。新的模板会继承当前模板所关联的所有模板上的监控元素，而直接创建在当前模板上的监控元素，则不会被复制给新创建的模板。这么说，你可能还是没有太明白是什么意思，下面举一个例子来加以说明。例如，我们当前正在配置的模板为 A，它除了从它所关联的模板 B 那里继承了 3 个监控项目外，其自身还有两个监控项目不是从任何模板那里继承过来的，而是直接添加在它自己身上的。所以，模板 A 总共有 5 个监控项目。这个时候，如果单击“复制”按钮，那么系统将会以模板 A 为基础，创建一个新模板 C。模板 C 会从模板 A 那里复制到模板的关联关系，即模板 C 也会关联模板 B。因此，模板 C 会从模板 B 那里继承过来 3 个监控项目。但是，模板 A 自身所创建的两个监控项目，不会被复制或传递给模板 C。因此，基于模板 A 复制过来的模板 C 总共有 3 个监控项目。

□ **完全复制 (Full clone)：**当单击这个按钮时，系统同样也会以当前模板为基础创建新的模板。但是，单击“完全复制”按钮所创建的新模板与前面介绍的通过单击“复制”按钮所创建的新模板的不同之处在于，单击“完全复制”按钮所创建的新模板，不但会从当前模板这里继续模板的关联关系，而且还将当前模板自身所拥有的监控元素也复制过去。依然以上面例子为例来加以说明。当在当前模板 A 的配置页面上单击“完全复制”按钮时，系统所创建的新模板 C 不但关联了模板 B，而且还会将模板 A 自身所拥有的两个监控项目的配置信息也复制过来，变成自己的监控项目。注意，我们这里所说的是将模板 A 自身的两个监控项目的配置信息复制过来，而不是继承。这两个监控项目的配置信息复制过来后，就变成了模板 C 自身的监控项目，它们跟模板 B 就没有任何关系了。因此，通过单击“完全复制”按钮所创建的新模板 C，总共会有 5 个监控项目。

□ **删除 (Delete)：**当单击这个按钮时，则系统将当前模板从系统中删除。这个按钮的功能与我们在前一节所介绍的下拉菜单项“删除选中项”是类似的。即，单击这个按钮，则只会将当前模板自身的信息从系统中删除，而关联当前模板的主机或模板，其上从当前模板上继续过去的监控元素，如监控项目、数据图、触发器等，不会因为当前模板的删除而删除。

□ **删除并清除数据 (Delete AND Clear)：**该按钮的作用与我们在前面章节中介绍的“删除选中项及关联元素”下拉菜单项的功能是相同的。即，如果单击这个按钮，则系统不但将当前模板的配置信息从系统中删除，而且将关联了当前模板的主机或模板所拥有的从当前模板继承过去的监控元素一并给删除掉。

模板通用信息配置表页面上还有另外两个选项卡：“关联模板”和“宏变量”。其中，“宏变量”选项卡是用来配置当前模板所拥有的模板级宏变量信息的。关于宏变量的配置方法，我们已经在前面章节中作过详细介绍，所以，在这里就不再重复。“关联模板”选项卡是用于配置当前模板所关联的模板的，单击这个选项卡，系统将显示出如图 5-3 所示的表单页面。

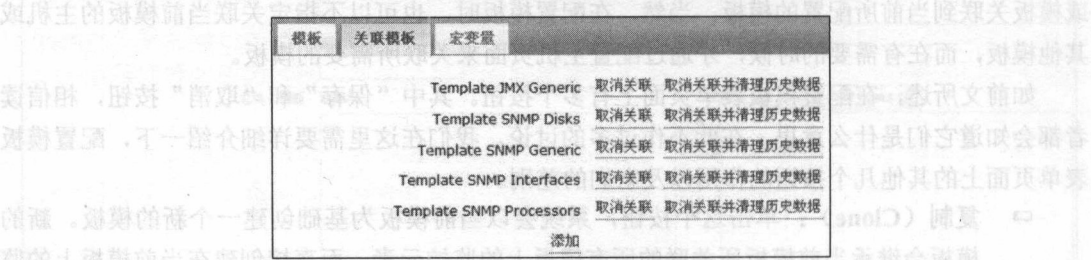


图 5-3 配置模板的关联模板表单截图

在前面的章节中多次提到过，在 Zabbix 系统中，模板本身也是可以关联其他模板的，这就是模板的嵌套。当把多个模板嵌套到一个模板上之后，再将这个关联了多个其他模板的模板关联到一台主机上，则该主机将会继承所有这些被嵌套模板上的监控元素。这么做至少有一个好处，那就是当有很多模板需要关联到主机上时，可以先将这些模板嵌套成少数几个模板，然后将这少数几个关联了其他模板的模板再关联到所需要的主机上，这样可以减小我们维护的工作量。

当需要在当前模板上关联其他模板时，只需在如图 5-3 所示的“关联模板”选项卡表单上单击“添加”超链接。然后，系统就会弹出一个模板选择窗口，在模板选择窗口中选中需要被关联的模板，并单击“选择”按钮，系统就会将我们所选中的模板添加到“关联模板”表单页

面上的对应列表中。而当单击“取消关联”超链接时，则可以取消当前模板对对应模板的关联。但是，这个操作仅取消模板之间的关联关系，它不会删除当前模板在被删除关联关系之前，从被关联模板那里继承过来的监控元素。这些之前从关联模板那里继承过来的监控元素的配置信息，会从前被关联的模板那里被复制过来，并且自动添加到当前模板上，就像这些监控元素是在当前模板上创建的一样。举个例子就可能更容易理解一些。例如，当前模板 A 关联了模板 B，而模板 B 有一个监控项目 a。因此，在删除模板 A 对模板 B 的关联之前，模板 A 上有一个监控项目 a。但是，这个监控项目不是直接在模板 A 上创建的，而是 A 通过 B 继承过来的。现在，当在模板 A 上使用“取消关联”超链接来删除与模板 B 的关联关系时，监控项目 a 不会因为这个关联关系的被删除从而从模板 A 上被删除掉。而是系统会自动将监控项目 a 的配置信息从模板 B 上复制过来，并将其添加到当前模板 A 上，就像监控项目 a 是直接在模板 A 上创建的一样。而当单击“取消关联并清理历史数据”超链接时，则系统会取消所对应的模板与当前模板之间的关联关系，并且会删除从所关联的模板那里继承过来的监控元素。所以，超链接“取消关联”和“取消关联并清理历史数据”的不同之处在于，单击“取消关联”超链接时，系统仅取消模板之间的关联关系，而保留之前从被关联模板那里继承过来的监控元素。而当单击“取消关联并清理历史数据”超链接时，则系统不但取消模板之间的关联关系，还会将之前从被关联模板那里继承过来的监控元素也一起删除掉，而且这些监控元素的删除是迭代进行的，即，所删除的监控元素如果之前通过当前模板继承给了主机或其他模板，则对应主机或模板上相应监控元素也会被一起删除掉。

为了下面的配置和测试能够顺利进行下去，也为了在下面章节中表述方便，我们这里创建一个新模板，模板名称为 test1。依次选择菜单项“系统配置”→“模板”，并在“配置模板”页面上单击“新建模板”按钮。然后，在打开的“配置模板”表单页面的“模板”选项卡里输入以下内容：“模板名称”表单项里输入 test1；“显示名称”表单项里输入 test1；已加入组选择并添加 Zabbix 监控服务器组，其他内容留空不用输入，如图 5-2 所示。“关联模板”选项卡上的表单内容不用填写。为方便后续我们测试宏变量功能，在这个新创建的模板上创建两个模板级的宏变量：{\$LOWDISKUSAGE} 值为 60；{\$HIGHDISKUSAGE} 值为 80，如图 5-4 所示。

宏变量	值	
{ \$LOWDISKUSAGE }	60	移除
{ \$HIGHDISKUSAGE }	80	移除
添加		

图 5-4 配置模板宏变量表单截图

完成上述配置后，单击“保存”按钮，保存我们对这个模板所作的配置。

5.1.3 关联模板到主机

虽然在每个模板上都可以创建诸如监控项目、触发器、数据图和图表等监控元素，但是，不能直接在模板上创建监控元素所需要使用到的数据采集接口。因此，我们不能单独使用模板来采集监控数据，更不能单独使用监控模板来触发动作报警。因此，模板必须至少要直接或间接地关联到一台监控主机，对应模板才有作用和意义。任何一个模板，如果它没有直接或间接地被关联到一台主机上，那么这个模板在系统中实际上是没有发挥作用的，因此，它实际上也就没有存在的必要性和意义了。所谓模板间接关联到主机上是指，一个模板 A 被关联到另一个

模板 B 上，而后模板 B 被关联到主机 C 上。那么就可以认为模板 A 是间接地被关联到主机 C 上了。

将一个模板关联到一台主机上有三种方法。方法一，是在创建一台新的主机时，通过新建主机表单页面上的“模板”选项卡，向新创建的主机上添加需要关联的模板。方法二，是通过修改一台已经存在主机的模板关联信息，将一个模板关联到一台已存在的主机上。操作步骤和方法是，依次选择“系统配置”→“主机”菜单项，在主机列表页面上单击主机名称上的超链接，然后在“配置主机”页面的“模板”选项卡上添加或修改要关联的模板。不管是在新建主机时添加关联的模板，还是通过修改已存在主机的模板关联信息来在主机上关联新的模板，在“配置主机”页面上都有如图 5-5 所示的“模板”选项卡。

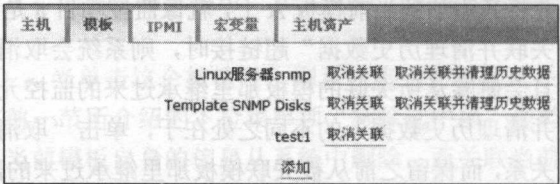


图 5-5 主机“模板”关联选项卡截图

图 5-5 所示的“模板”选项卡上的各类超链接的用法和功能，与图 5-3 所示的“关联模板”选项卡上对应超链接的用法和功能是类似的。即，单击模板信息后面的“取消关联”超链接，则表示取消对应模板与对应主机的关联关系，但是保留该主机从被删除关联模板上所继承的监控元素。而单击“取消关联并清理历史数据”超链接，则表示删除对应主机与对应模板之间的关联关系，且删除该主机从对应模板上继承过来的监控元素。而当单击“添加”超链接时，则可以在当前主机上添加新的模板关联关系。

将一个模板关联到一台主机的第三种方法是，在图 5-2 所示的“配置模板”页面的“模板”选项卡上，在“主机/模板”多选框组的“已加入”多选框中添加上需要被关联的主机，即可将对应模板关联到所选择的主机上。

如前文所述，一个模板只有至少被关联到一台主机，该模板才会发挥作用。所以，在下面的介绍中，如果你需要测试添加在模板上的监控元素的效果，则请按照上述的方法，将对应模板关联到一台主机上。

5.2 配置监控项目

在上一节中，介绍了如何在 Zabbix 系统中配置一个新的监控模板。但是，当时只介绍了如何配置监控模板的通用信息，并未向新创建的模板上添加任何监控元素。所以，在前一节，实际上我们只是创建了一个没有任何监控元素的空模板。而如果一个监控模板没有任何监控元素，则这个监控模板即使被我们关联到一台主机上，它也将是毫无用处，没有任何实际意义的。所以，在接下来的章节中，我们将详细介绍如何在一个模板上配置监控元素，比如，监控项目、数据图、触发器和低级自动发现规则，等等。

实际上，在 Zabbix 系统中，在一个模板上配置监控元素的操作过程和操作步骤，与在一台主机上配置相应监控元素的操作过程与操作步骤并没有太大的差别。最大的差别仅在于，在主机上配置监控项目时需要指定监控数据采集接口，而在模板上配置监控项目时则不需要指定监

控数据采集接口。而这仅有的一点差别，相信读者应该能理解和掌握。因此，在接下来的章节中所介绍的配置监控元素的方法，既可以将其运用在主机上相应监控元素的配置上，也可以将其运用在模板上相应监控元素的配置上。而如果将监控元素配置在一个模板上，则只需将对应的模板关联到主机上，系统便能自动在对应的主机上创建相应的监控元素。我们这里，将会将这些监控元素配置在刚刚创建的 test1 模板上。同时，本书将不再单独介绍在主机上配置监控元素的方法和过程。不过，实际上已经在第3章介绍过如何在一台主机上配置监控元素了，而且还给出了在一台主机上配置监控元素的操作方法和操作过程。

5.2.1 配置获取主机硬件信息的监控项目

通过前面章节的介绍，我们已经了解到，在 Zabbix 系统中，有一类监控项目，它们可以通过 Zabbix 系统的被监控设备代理组件来采集所需要的监控数据。这里，就在我们刚刚创建的模板 test1 上，创建一个这类监控项目。将要配置的监控项目是一个采集主机硬件信息的监控项目，它的监控数据中包含主机供应商（品牌）信息、硬件型号和序列号等，这些信息是通过被监控设备代理组件读取被监控设备内存中的信息获取的。因此，为了保证该监控项目能够正确地采集到监控数据，在配置该监控项目之前，需要按照前面章节中所介绍的方法和步骤，在被监控设备上安装和配置好 Zabbix 系统的被监控设备代理组件，并启动该组件进程。同时，如上面所述，被监控设备代理组件是通过读取被监控设备内存中的信息，从而获取被监控设备的硬件信息的。所以，需要让被监控设备代理组件具有读取相关内存信息的权限。而要让被监控设备代理组件具有读取相关内存信息权限的方法是：修改被监控设备代理组件的配置文件 zabbix_agentd.conf，将其中的 AllowRoot 配置项的值修改为 1，并重启 zabbix_agentd 进程。

查看本书的附录 C，可以很容易知道，在 Zabbix 系统中，通过被监控设备代理组件采集监控数据的监控项目有多达五六十种。在这里，我们没有办法对这五六十种监控项目都一一举出配置实例。所以，如果要熟练掌握这类监控项目的配置，并对它们的配置有深刻的认识和掌握，还需自行多做一些测试和研究。

在一个模板上添加新的监控项目的方法很简单，依次选择菜单项“系统配置”→“模板”，在“配置模板”列表页面上，找到刚刚新创建的模板 test1，并单击对应行的“项目”超链接。然后，单击“配置项目”列表页面右上部的“新建项目”按钮，系统将打开“配置项目”表单页面，如图 5-6 所示。

在图 5-6 中，依次在“名称”栏里输入服务器硬件信息；“类型”表单项里选择“Zabbix agent”选项；key 栏里输入“system.hw.chassis[info]”或单击“选择”按钮，并从系统弹出的窗口里选择“system.hw.chassis[<info>]”项，同时将对对应关键字的参数修改成“info”；而“信息类型”表单项里选择“文本”；“更新间隔（秒）”栏里输入 3600。因为我们当前添加的监控项目是用于采集服务器硬件信息的，一般来说，服务器的硬件信息不会频繁变动，即使有所变动，对于服务器的运行也不会构成重大的影响。所以，出于减小 Zabbix 系统数据库压力方面的考虑，我们在这里将“更新间隔”调整为一小时一次。实际上，如果有必要，我们还是可以延长更新间隔的，比如将其调整为 43200 或 86400，即每半天或一天更新一次。同样是出于减小 Zabbix 系统数据库压力方面的考虑，在这里我们将“保留历史数据（天）”表单项的内容调整为其可接受的最小值，即 1 天。因为，一般来说，我们比较关注的是，服务器当前的硬件配置情况，而对于服务器硬件在历史上是否做过变动，并不太在意。诚如前文所述，我们当前所配置的监控项目是用于采集主机的硬件信息的，所以，为便于管理，我们在“新建监控分类”栏

里输入“硬件信息”，以便系统在创建当前监控项目时，自动创建一个新的“硬件信息”分类，并将新创建的监控项目加入到该分类中。

图 5-6 配置监控项目表单页面截图

完成上述配置后，单击页面上的“保存”按钮，系统便会在 test1 模板上创建一个名称为“服务器硬件信息”的监控项目。之后，当将这个模板关联到某台主机上，并在对应主机上安装配置好被监控设备代理组件，同时启动它后，该主机上的“服务器硬件信息”监控项目不久便会采集到如“Dell Inc. PowerEdge R610 XXXXXXXX Rack Mount Chassis”形式的服务器硬件信息。

如上所述，我们在 test1 模板上配置了一个通过被监控设备代理组件采集监控数据的监控项目。诚如我们所了解的那样，这类监控项目有多达五六十种之多。所以，如果要熟练掌握这类监控项目的配置，并对它们有深刻的认识和掌握，还需要多多做一些实际测试和研究。在监控一台主机时，配置这种类型监控项目的一个最大好处就是，无须额外编写监控数据采集脚本程序，而只要在被监控主机上安装、配置好被监控设备代理组件并启动它即可。

5.2.2 配置 Web 端口状态监控项目

当需要监控某个服务的状态时，一般只需监控这个服务所对应端口的状态即可。此时，如果在 Zabbix 系统中配置相应的“简单检查”类监控项目，或许是满足这类监控需求的最简单也是最方便的方法。因为，“简单检查”类监控项目无须在被监控主机上安装和配置任何额外的软件，同时，也不需要我们额外编写任何脚本程序，而只需我们按照要求配置相应的监控项目即可。

在通过“简单检查”方法采集指定服务的端口状态时，对于一些常见的服务，我们在配置监控项目时，只需在其关键字参数里指定对应服务的名称，如 http、pop、smtp 等即可，而无须指定对应服务的端口号。此时 Zabbix 服务器端（或其代理端）组件将使用对应服务的标准端口号作为目标端口号，连接被监控主机上相应的端口。当然，如果所监控的这些常见的服务，其

所使用的端口是非标准的端口号（或者说是非对应服务的默认端口号），则对应监控项目关键字的服务名称和端口号参数都需要指定。或许，你跟我一样，在刚接触“简单检查”这种监控数据采集方法时，产生了一些疑惑：既然这种类型监控项目的关键字参数 `service` 可以是 `tcp`，而不管是 `http` 服务、`pop` 服务还是 `smtp` 服务等，它们在通信时，其网络层的协议使用的都是 `tcp` 协议。既然如此，在简单检查时为什么服务名称不都使用 `tcp`，而是还需要区分 `http`、`pop`、`smtp` 等服务呢？这是因为，我们知道，不同的服务，当有客户端连接到它所侦听的端口时，它们给客户端返回的信息是不一样的。例如，当有一个客户端连接 `HTTP` 服务所对应的 `80` 端口时，它是不给客户端返回任何信息的，只有客户端执行某一请求时，相关进程才会给客户端返回与该请求相关的信息；而当有客户端连接 `POP3` 服务的端口时，则只要连接建立完成，服务端都会给客户端返回“OK”字样的信息；而 `SMTP` 服务返回给客户端初始连接的信息是 `220` 状态码。所以，对于这些常见的服务，Zabbix 服务器通过匹配这些服务所返回给它的信息，来判断对应的服务是否是正常的，这将使判断的结果更准确。关于这一点，如果研究了 Zabbix 系统的源码，就会很清楚 Zabbix 系统为什么要这么做。

另外，需要提醒读者注意的是，简单检查类监控项目关键字的 `service` 参数是大小写敏感的。Zabbix 系统只接受小写的 `service` 参数。如果在配置监控项目时，不小心将这个参数内容写成了大写形式，虽然在 Zabbix 系统的 Web 前台页面上是没有任何的提示和警告的，但是，因为 Zabbix 系统的服务器端组无法识别出声称的服务，所以，对应监控项目将会变成不被支持的监控项目。

接下来在 `test1` 模板上，再创建一个采集 Web 端口状态的监控项目。配置方法与我们在前面介绍的“配置获取主机硬件信息的监控项目”的配置方法是类似的，你只需在如图 5-6 所示的表单页面上输入或选择如下表单项的内容，并单击页面下部的“保存”按钮，即可创建相应的监控项目：

- 名称。输入“Web 端口状态”。
- 类型。选择“Simple check”选项。
- Key。输入“`net.tcp.service[http]`”，或单击表单项后面的“选择”按钮，并选择“`net.tcp.service[service,<ip>,<port>]`”选项，且将关键字参数修改成 `[http]`。
- 信息类型。选择“数值（无符号整型）”。
- 更新间隔（秒）。60。
- New flexible interval。该行的“间隔(秒)”表单项里输入 30，“周期”表单项里输入“1-5,09:00-18:00”，并单击其后的“添加”按钮。
- 保留历史数据（天）。0。
- 保留趋势数据（天）。0。
- 显示值。选择“Service state”项。
- 新建监控分类。`HTTP 服务`。

我们知道，一般来说，Web 服务在白天的访问量要比晚上的访问量大一些，所以白天如果服务出现故障，则其影响也会更大一些。所以，在上面这个监控项目中，我们配置了一个“可变更新闻隔”，指定在每个星期的星期一到星期五的每天 9:00~18:00 期间，该监控项目的监控数据采集间隔为 30 秒，而其余时间更新间隔为 60 秒。类似的，因为我们配置这个监控项目的目的是监控被监控主机上的 Web 端口是否正常，如果不正常，则立即给我们发送报警信息。所以，这个监控项目对于我们来说，主要是用于报警使用的。而我们对这个监控项目的历史数据以及历史数据的变化趋势并不太关心。所以，在这里，我们将这个监控项目的“保留历史数据

“（天）”和“保留趋势数据（天）”两个配置项都配置成了 0。只是，将这两个配置项都配置成 0 时，引用该监控项目的触发器表达式里只能使用 last 函数，且函数的参数只能用时间 0，而不能使用表示次数的数字。在后续章节中，将会介绍如何为我们在这一章所配置的这个监控项目，配置一个触发器。而因为我们所配置的监控项目，其将来所采集的监控数据的原始值是 0 或 1 这两个表示状态的数字，为了使状态数字在显示或报警时，显得更人性化一些，在这里我们选择使用了“Service state”这个数据映射。该数据映射是 Zabbix 系统预配置好的，所以只需使用即可。

5.2.3 配置 Nginx 状态数据监控项目

我们知道，在 Web 服务器 Nginx 中有一个状态模块，当开启这个模块后（要开启 Nginx 的这个功能模块，需要在编译 Nginx 时显式地带上--with-http_stub_status_module 编译选项，且在 Nginx 配置文件里做好相应的配置），可以通过浏览器查看到 Nginx 进程的状态数据。这些状态数据包括：当前活动连接数、Nginx 从上次启动以来总共处理的连接数、Nginx 从上次启动以来总共创建的握手次数、Nginx 从上次启动以来总共成功处理了的请求数等信息。在这里，我们来介绍一下如何在 Zabbix 系统中，通过外部脚本程序的方式采集 Nginx 状态模块中所输出的这些状态数据。

既然叫作“通过外部脚本程序采集 Nginx 状态数据”，那么首先要做的就是编写这个用于采集监控数据的脚本程序。这里用 Python 语言编写了这个脚本程序，其源代码如下：

```
1. #!/bin/env python
2. #coding:utf-8                                #设置程序文件的编码为 utf8 编码
3. try:
4.     import urllib2                            #引入 urllib2 工具包
5.     import sys                                #引入 sys 工具包
6.     import string                             #引入 string 工具包
    #如果上述引入工具包出错，打印-1 并退出
7. except:
8.     print -1
9.     sys.exit(-1)
10.
11.
12. def isNum(value):                            #定义一个函数 isNum
13.     try:
14.         int(value) + 1
15.     except Exception,e:
16.         return False
17.     else:
18.         return True
19. numList = []                                #定义一个列表
    #如果调用该脚本时所输入的参数小 2 个，则退出并返回退出码-1
20. if len(sys.argv)<3:
21.     print -1
22.     sys.exit(-1)
23. try:
24.     if string.strip(sys.argv[2]) != None:    #如果第二个参数内容不为空
        #拼出所需要打开的 URL 连接地址
25.         url="http://"+string.strip(sys.argv[1])+"/nginx_status"
26.         fp=urllib2.urlopen(url)            #打开指定的 url 页面
```

```

27.     line=fp.readlines()
28.     for content in line:                                #从页面上读取所有内容
29.         if content.strip():
30.             for num in content.split(' '):
31.                 if isNum(num):
32.                     numList.append(num)
33.     else:
34.         print -1
35.         sys.exit(-1)
36.     if len(numList)<7:
37.         print -1
38.         sys.exit(-1)
# 以下是匹配 Nginx 状态页面上的各字段值, 并打印出它们
39.     if sys.argv[2].strip().lower() == "active_connections":
40.         print numList[1]
41.     elif sys.argv[2].strip().lower() == "accepts":
42.         print numList[2]
43.     elif sys.argv[2].strip().lower() == "handled":
44.         print numList[3]
45.     elif sys.argv[2].strip().lower() == "requests":
46.         print numList[4]
47.     elif sys.argv[2].strip().lower() == "reading":
48.         print numList[5]
49.     elif sys.argv[2].strip().lower() == "writing":
50.         print numList[6]
51.     elif sys.argv[2].strip().lower() == "waiting":
52.         print numList[7]
53.     else:
54.         print -1
55.         sys.exit(-1)
56. except Exception,e:
57.     print -1
58.     sys.exit(-1)

```

接下来, 需要将上述这段 Python 源代码输入到一个文件中, 并将该文件保存为 `get_nginx_status.py` 文件。同时, 将该脚本文件上传到 Zabbix 服务器的 Zabbix 安装目录下的 `share/zabbix/externalscripts/` 目录下。如果你是按照我们在前面章节中所介绍的方法编译安装 Zabbix 服务器端组件的, 则这个脚本程序文件的存放路径就是 `/opt/zabbix/share/zabbix/externalscripts/get_nginx_status.py`。最后, 需要执行以下两条命令, 以便给该脚本程序文件赋予可执行权限, 并将它的属主和属组设置为 `zabbix` 用户和 `zabbix` 用户组, 从而使 Zabbix 服务器端进程能够正确地调用它。

```

shell> chmod +x /opt/zabbix/share/zabbix/externalscripts/get_nginx_status.py
shell> chown zabbix:zabbix /opt/zabbix/share/zabbix/externalscripts/get_nginx_status.py

```

在通过 Web 页面添加监控项目之前, 请确认 Zabbix 服务器端组件配置文件 `etc/zabbix_server.conf` 中 `ExternalScripts` 配置项所配置的路径是否正确。如果这个配置项中所配置的路径并不是脚本程序文件 `get_nginx_status.py` 的存放路径, 则请将其配置成相同的路径, 并重启 `zabbix_server` 进程。

上面我们给出的脚本程序的调用方法是 `<PATH_TO_SCRIPTS>/get_nginx_status.php <主机名> <字段参数>`。其中主机名可以是被监控主机的 IP 地址, 也可以是被监控主机的 DNS 主机

名。而字段参数,则用于指定 Nginx 状态信息中某个字段,它们可以是 active_connections、accepts、handled、requests、reading、writing、waiting 等中的任何一个。而不管是“主机名”还是“字段参数”,都会在下面配置对应监控项目时,通过监控项目关键字参数来指定它,并在 Zabbix 服务器端进程调用这个脚本程序时,通过 Zabbix 服务器端进程传递给脚本程序。当这个脚本程序运行出错时,则会给 Zabbix 系统返回数值-1,否则返回获取到的指定字段的 Nginx 状态数。

完成上述配置后,接下来通过 Zabbix 系统的 Web 前端在 test1 模板上添加一个监控项目,以用于采集 Nginx 状态数据。配置 Nginx 状态监控项目的方法与前面所介绍的配置 Web 端口状态监控项目的方法是相似的,只需注意以下表单项内容需要按要求填写,其他表单项内容你可以根据自己的需要填写:

□ **类型**。选择“External check”选项。

□ **Key**。填写“get_nginx_status.py[\"{HOST.NAME}\",\"active_connections\"]”。

其中,监控项目关键字参数{HOST.NAME}是 Zabbix 系统中预定义的全局宏变量,在这里你也可以填写成{HOST.IP}、{HOST.CONN}或{HOST.DNS}等。但是,不管你使用哪个宏变量,只需保证从 Zabbix 服务器上,通过 http://{YMACRO}/nginx_status 地址形式能够访问到被监控主机上 Nginx 状态模板页面即可。而参数“active_connections”,则用于指定获取 Nginx 状态页面上的某个字段的内容。

通过分析脚本程序,不难看出,当使用外部脚本程序采集监控数据时,Zabbix 系统实际上是捕获脚本程序执行后输出的内容作为被监控项目上所采集到的监控数据。因此,如果脚本程序在被调用时出错,抛出出错信息,则该出错信息也将会被视作所采集到的监控数据看待,Zabbix 系统是无法判别脚本程序在执行过程中有没有出错的。因此,当需要使用这种方法采集监控数据时,一般要求脚本程序里有异常处理功能,以避免因为 Zabbix 系统采集到意外数据(比如,将监控项目的信息类型配置成数值型,而脚本程序在出错时很可能会抛出很长的文本型信息),而导致监控项目不被支持。

最后,如果不是必需的,不建议使用外部脚本程序的方法采集监控数据,因为 Zabbix 服务器需要定期地调用数据采集脚本,而这会严重影响 Zabbix 服务器的性能。

5.2.4 配置数据库监控项目

通过前面章节的介绍我们知道,所谓数据库监控项目是指 Zabbix 服务器端进程通过 ODBC 协议,直接查询被监控主机上的数据库目标表,从而采集所需要的监控数据的监控项目。因此,这类监控项目可以较高效地采集到应用程序的监控数据,而且可以较容易地采集到与我们自己个性化业务相关的监控数据。

在具体配置数据库监控类监控项目之前,除了要按照前面章节中介绍的方法在 Zabbix 服务器端或 Zabbix 服务器代理端上安装并配置 unixODBC 软件包和相关驱动程序外,还需要配置一个用于连接目标数据库的数据源。配置数据源的方法很简单,只需修改/etc/odbc.ini 配置文件,向该文件中添加上相应的配置内容即可。如将下面这段配置信息添加到这个文件中并保存,就配置好了一个 Oracle 数据源。

```
1. [Oracle]
2. driver=Easysoft ODBC-Oracle WP
3. description=Easysoft Oracle ODBC WP driver
4. server=192.168.5.139
5. port=1521
```



```

6. sid=
7. user=zabbix
8. password=zabbix
9. logging=No
10. logfile=/var/log/oracle_driver.log
11. enable_user_catalog=yes
12. enable_synonyms=yes
13. metadata_dont_change_case=no
14. metadata_dont_do_schema=no
15. metadata_id=no
16. limit_long=0

```

上面这段配置中，“[Oracle]”为数据源的名称，这个名称在配置监控项目时会使用到，且不得有重复。第 2 行，指定该数据源使用哪个驱动程序，而各类数据库的驱动信息在/etc/odbcinst.ini 配置文件中配置。第 4 行，指定被监控数据库的 IP 地址或主机名，一般也就是监控项目上所配置的主机 IP 地址或主机名。接下来的 port、sid、user、password 等配置项，则为连接目标数据库所需要的端口、服务 ID、用户名和密码等参数。当使用上述配置的数据源配置一个监控项目，用于统计业务系统日新注册用户数时，则“配置项目”表单页面上各表单项的内容只需按下列内容输入或选择即可：

- ❑ 名称。日新注册用户数统计。
- ❑ 类型。选择“Database monitor”选项。
- ❑ Key。db.odbc.select[user_count]。
- ❑ 附加参数。

```

DSN=Oracle
sql=select count(*) from users

```

- ❑ 更新间隔（秒）。86400。
- ❑ 取值方式。选择“增量（变化）”选项。

在“附加参数”表单项中指定当前监控项目所使用的数据源、查询数据所使用的 SQL 语句等。需要注意的是，这两行内容必须分行来书写，如果写在一行，则 Zabbix 系统会报错。因为这个监控项目是用户统计日新注册用户数的，所以，采集间隔就需要写成一天的总秒数，即 86400。上面所写的 SQL 语句，查询目标数据库中 users 表中总的用户数，所以如果要统计每日新增的用户数，取值方式就需要选择“增量（变化）”选项。

提示：1. Easysoft 提供的各种数据库的 ODBC 驱动程序正式的 License 是要收费的。试用 license 有效期只有一个月。所以，如果需要长期采用 ODBC 技术采集监控数据，需要向 Easysoft 购买正式的许可证书。

2. 通过我们测试和 Zabbix 官网 Buglist 上所提供的信息，当使用数据库监控项目采集监控数据时，如果被采集数据的目标数据库是 MySQL，则可能会引起 Zabbix 系统服务器端进程异常退出。该 Bug 在 Zabbix 系统的多个版本如 2.0、2.2.0 中存在。

5.2.5 配置磁盘读取速率监控项目

在前面介绍监控数据采集方法时介绍过，通过 SSH 协议采集监控数据的原理就是，Zabbix 服务器端或其代理端根据配置，使用 SSH 协议连接到被监控主机上，然后在被监控主机上执行

用户配置的命令序列或者脚本程序，Zabbix 服务器端或其代理端捕获命令序列或脚本程序所返回的结果，作为监控项目所采集到的监控数据。因此，通过 SSH 协议采集监控数据与通过外部脚本采集监控数据的根本区别在于，通过 SSH 协议方法采集监控数据时，被执行的命令序列或脚本程序是在被监控主机上执行的，而通过外部脚本程序采集监控数据时，被执行的命令序列或脚本程序是在 Zabbix 服务器或其代理服务器上执行的。所以，这两种监控数据采集方法所执行的命令序列或脚本程序的环境与宿主都是不同的。

然而，与通过外部脚本程序采集监控数据不同，通过 TELNET 协议采集监控数据与通过 SSH 协议采集监控数据，在本质上是一致的，只是在采集监控数据时，Zabbix 服务器与被监控主机之间通信所采用的通信协议不同而已。当然，因为通信协议不同，所以验证方法也稍有不同（TELNET 协议不支持公钥私钥方法验证）。故此，在这里我们主要介绍如何配置通过 SSH 协议采集监控数据的监控项目。而对于通过 TELNET 协议采集数据的监控项目，如果读者感兴趣，可以对比我们在这里介绍的如何配置通过 SSH 协议采集监控数据类监控项目的方法，自行研究，应该不难于配置。

在前面介绍 Zabbix 系统中监控数据采集方法时介绍过，要使 Zabbix 系统支持通过 SSH 协议采集监控数据，则需要在 Zabbix 服务器或其代理上安装 libssh2 软件包，同时在编译安装 Zabbix 服务器端组件或其代理端组件时，需要显式地使用 `--with-ssh2` 编译配置选项，以使 Zabbix 服务器端组件支持使用 SSH 协议连接被监控主机。

我们知道，对于一个正在运行着的系统，如果能实时采集并监控该系统中各块磁盘的读写速率，也即磁盘 IO 速率，对于了解和掌握系统中各块磁盘的读写性能，识别和判断系统性能瓶颈，进而为系统调优提供重要的数据参考，都有非常重要的意义。下面就来配置这样一个监控项目，用它采集和监控系统中 sda 磁盘的读取速率。要创建磁盘读取速率监控项目，请在“配置项目”表单页面上按照下面的内容选填各表单项的内容：

- 名称。sda 磁盘 IO 读取速率。
- 类型。SSH agent。
- Key。ssh.run[sda_readspeed,,2228]。
- Authentication method。密码。
- 用户名。root。
- 密码。password。
- 已执行脚本。`/usr/bin/iostat -k |/bin/grep "sda" |usr/bin/head -1 |usr/bin/awk -F " " '{print $3}'`。
- 信息类型。数值（浮点数）。
- 单位。kbs。

以上配置中，如果“Authentication method”表单项选择“公钥”，则在配置监控项目之前，需要按照我们在前面章节中所介绍的方法，在 Zabbix 服务器和被监控主机之间配置好公钥私钥验证方式。如果在生成公钥私钥时，使用了 passphrase，则在配置监控项目时，Key Passphrase 表单项也要输入相应的密码。

有几点需要特别指出说明的是：

第一、在前面介绍过，当通过 SSH 协议采集监控数据时，监控项目上所配置的命令序列或脚本程序实际上是在被监控主机上运行的。所以，监控项目上所配置的命令序列中所使用到的命令在被监控主机上必须是存在的，且对于连接被监控主机的用户则需要有执行相应命令序列

的权限。例如，当将上述配置有“sda 磁盘 IO 读取速率”监控项目的模板关联到某台被监控主机上，比如 192.168.5.137 上时，则主机 192.168.5.137 上必须要存在 iostat、grep、head、awk 等命令，且这些命令的路径要与上面所配置的路径一致。否则，系统将无法正确采集到需要的数据。

第二、通过 SSH 协议采集监控数据的监控项目，其关键字是具有固定格式的，它的格式是：`ssh.run[<unique short description>,<ip>,<port>,<encoding>]`。其中，参数 `unique short description` 是该监控项目在 Zabbix 系统中的唯一标识符，该参数内容可以任意指定，只需满足 Zabbix 系统中关于关键字的命令规范即可；参数 `ip` 则用于指定被监控主机的 IP 地址或主机名，如果不填写，则系统将以监控项目所在的主机的 IP 地址作为该参数的值；`port` 即为被监控主机上运行的 SSH 服务所侦听的端口号，默认值是 22，如果被监控主机上 SSH 服务所侦听的端口是默认端口号，则该项可以不用填写；而参数 `encoding` 则用于指定编码信息。

第三、通过 TELNET 协议采集监控数据的监控项目，其关键字也是具有固定格式的，它的格式是：`telnet.run[<unique short description>,<ip>,<port>,<encoding>]`。该关键字中各参数的含义，与通过 SSH 协议采集监控数据所对应的监控项目的关键字中相应参数的含义是相同的，在此就不再一一复述。TELNET 服务的默认端口号是 23。

5.2.6 配置 Tomcat 性能监控项目

通过 Java 管理扩展（Java Management Extensions, JMX）采集 Java 应用和 Java 容器的监控数据，可能是 Zabbix 系统对 Java 应用和 Java 容器进行监控时，最方便也是最高效的监控数据采集方法。在前面介绍监控数据采集方法时，我们已经介绍过，要想让 Zabbix 系统能够通过 JMX 方法采集监控数据，则需要在 Zabbix 系统的服务器端安装 Java 环境，并且在编译安装 Zabbix 服务器端组件时，需要使用 `--enable-java` 编译配置选项，使 Zabbix 系统能够支持通过 JMX 方法采集监控数据。当 Zabbix 系统支持通过 JMX 方法采集监控数据时，则在 Zabbix 系统安装路径的 `sbin` 目录下会有一个 `zabbix_java` 目录。该目录下存放的是 Zabbix 系统的 Java 应用程序网关组件。当需要让 Zabbix 系统通过 JMX 方式采集监控数据时，则需要启动这个 Java 应用程序网关组件。启动方法是执行 `<Zabbix_Install_Path>/sbin/zabbix_java/startup.sh` 启动脚本程序。

按照在前面章节中所介绍的方法，完成对被监控主机上 JMX 和 Zabbix 服务器端 Java 应用程序网关的配置后，还需要通过 Zabbix 系统的 Web 前端组件，针对被监控主机配置相应的 JMX 接口，相应的被监控主机上的 JMX 类监控项目才能正常地采集到监控数据。该接口的配置方法很简单，IP 地址或主机名毫无疑问就是被监控主机的 IP 地址或主机名。不过需要注意的是，这里所配置的 IP 地址或主机名要与被监控主机在启动 RMI 服务时，使用 `Djava.rmi.server.hostname` 参数所指定的 IP 地址或主机名保持一致，否则 Zabbix 系统的 Java 应用程序网关组件将无法连接到被监控主机的相应服务端口。而 JMX 接口中的端口则是被监控主机上 RMI 服务所侦听的端口，这个端口一般在启动被监控主机上的 RMI 服务时，通过 `-Dcom.sun.management.jmxremote.port` 参数来指定。

在 Zabbix 系统的 Web 页面上，配置通过 JMX 方法采集监控数据的监控项目，其方法与配置其他类型的监控项目没有什么太大差别。这里简单介绍一下，在 Zabbix 系统中，配置一个采集 Tomcat 容器 8080 端口每 ns 所处理的请求数的监控项目。请按照前面介绍的操作方法，在 `test1` 模板上创建一个新的监控项目，该监控项目在监控项目配置表单页面上的部分表单项内容请按以下内容进行填写：

- 名称。8080 端口每 ns 请求数。
- 类型。JMX agent。
- Key。jmx["Catalina:type=GlobalRequestProcessor,name=http-8080,requestCount"]。
- 用户名。monitor。
- 密码。snmp@domain.com。
- 信息类型。数值（浮点数）。
- 取值方式。增量（变化/每秒）。

其中，表单项“Key”为监控项目的关键字。JMX 类型监控项目的关键字是有固定格式的，它的格式是 `jmx[<object name>,<attribute name>]`。关于这个关键字中参数的含义和命令规范，请参阅我们在前面章节中所作的介绍和说明。而上述要求填写的“用户名”和“密码”，则是在被监控主机上运行的 RMI 服务中配置和指定的。Zabbix 系统的 Java 应用程序网关组件将会使用这里配置的用户名和密码，连接被监控主机上的 RMI 服务，并采集所需要的监控数据。因此，如果这里配置的用户名和密码不正确，则 Zabbix 系统将无法采集到所需要的监控数据。关于被监控主机上运行的 RMI 服务的用户名和密码的配置方法，请参阅我们在前面章节中介绍 JMX 数据采集方法时所作的介绍。而取值方式之所以选择“增量（变化/每秒）”选项，是因为这里配置的监控项目，其采集的原始数据是 Tomcat 自启动以来，从 8080 端口所接收到的总的请求数。而这里这个监控项目所要求的数据是每 ns Tomcat 8080 端口所接收到的请求数。该监控项目“配置项目”表单页面上的其他表单项内容，你可以根据自己的需要进行填写。

填写完成之后，单击页面下部的“保存”按钮，将所配置的监控项目的配置信息保存到系统中，至此监控项目就创建完成了。之后，将 test1 模板关联到需要的被监控主机上，如果所有配置都是正确的，则该监控项目就可以正常采集监控数据了。

5.2.7 配置 IPMI 监控项目

前面介绍过，通过 IPMI 方法可以采集到被监控主机上的硬件状态信息，例如，CPU 温度、CPU 风扇转数等。而且，我们提到过，通过这个方法采集监控数据时，不受被监控主机操作系统性能和状态的影响，也不受被监控主机是否已开启的影响。同样，要被配置为 IPMI 类型的监控项目，则被监控主机上需要添加 IPMI 接口。需要说明的是，通过 IPMI 方法采集监控数据时，被监控主机与 Zabbix 服务器之间的通信不是通过被监控主机上普通网卡进行的，而是通过被监控主机上专用的管理接口进行的。这些管理接口，在不同品牌服务器上的叫法不一样，Dell 服务器将其称之为 DRAC 接口，HP 服务器将其叫作 iLO 接口，而 IBM 服务器把它叫作 RSA 接口。因为在被监控主机上采集 IPMI 类型监控项目的监控数据与采集其他类型监控项目的监控数据所使用的网络接口不一样，所以，即使是同一台被监控主机，IPMI 类型接口所对应的 IP 地址，与其他类型数据采集接口所对应的 IP 地址一般是不一样的。

配置 IPMI 类型监控项目的方法，与配置其他类型监控项目的方法是类似的。这里，我们在 test1 模板上创建一个监控 Dell PowerEdge R410 服务器温度的监控项目。该项目在“配置项目”表单页面上的部分表单项的内容请按以下内容进行选填：

- 名称。服务器温度。
- 类型。IPMI agent。
- Key。ipmi.sensor.Ambient_Temp。

- ❑ IPMI 传感器。Ambient Temp。
- ❑ 信息类型。数值（浮点数）。
- ❑ 单位。C。

其中，需要对“IPMI 传感器”稍加说明一下，其他表单项的内容很好理解，不需要过多说明。其实，传感器的概念也很好理解。我们知道，想要查看和了解主机的某个硬件状态，例如 CPU 的温度、CPU 风扇当前转速、服务器温度等，就需要有一定的电子设备及相应的驱动程序将这些硬件状态信息读取并转化出来，然后呈现给我们。完成上述这部分工作的部件及驱动程序，就是我们所说的传感器。因为对于一台主机来说，硬件状态信息很多，包括 CPU 温度、当前电源状态、CPU 风扇转速、当前系统电流大小、当前系统风扇转速，等等。所以，对于我们所要监控的主机，传感器不可能只有一个，而是有许多个传感器。要查看一台主机中所有传感器的状态信息，则可以使用下列命令：

```
shell> ipmitool sensor
```

5.3 正则表达式及低级自动发现规则配置

做过程序开发或系统运维的朋友，大概对正则表达式都不陌生。在 Zabbix 系统中，可以通过 Web 前端页面定义正则表达式。而正则表达式一旦被定义，就可以在 Zabbix 系统的多个地方使用它。例如，通过正则表达式来匹配低级自动发现功能所发现的潜在监控对象；在监控项目的关键字中，也可以使用正则表达式来匹配所采集到的监控数据。

5.3.1 正则表达式介绍

正则表达式 (regular expression) 是对字符串进行操作的一种逻辑公式，其就是一种用事先定义好的一些特定字符以及这些特定字符的组合，组成一个“规则的字符串”，然后用这个“规则的字符串”来过滤或匹配目标字符串的操作。从上面关于正则表达的定义中，我们不难看出，正则表达式本身也是一种字符串，只是这是一个能表达某个规则的字符串。既然正则表达式这种字符串是用来表达某种规则的，那么组成一个正则表达式的字符串中肯定需要包含一些具有某种特殊意义的字符或字符串，否则无法通过一个普通意义上的字符串来表达某种规则。那么，这些在正则表达式中具有特殊意义的字符就叫作元字符。

Zabbix 系统支持完整的可移植操作系统接口 (Portable Operating System Interface for UNIX, POSIX) 扩展的正则表达式 (POSIX extended regular expressions)。可移植操作系统接口扩展的正则表达式，是正则表达式三种引擎中的一种，它广泛应用在 Linux、UNIX 等 UNIX 和类 UNIX 系统中。例如，Linux 系统中 grep、awk 等命令工具，就使用这种正则表达式引擎。下面，我们介绍几个在日常维护监控系统的过程中，可能需要使用到的正则表式的元字符。

表 5-1 正则表达式元字符列表

元字符	描述
. (点号)	用于匹配字符串中各种打印或非打印字符，但是换行符除外。例如字符串aac、abc、acc、adc等就可以被正则表达式a.c匹配成功。但是，字符串aabc、cabdbc等字符串就不能被正则表达式a.c匹配成功。通俗一点说，(.)元字符在进行正则表达式匹配时，它可以代表任何单个字符

续表

元字符	描述
[] (中括号)	表示使用被中括号括起来的字符集，来匹配目标字符串中单个字符。例如，当使用正则表达式a[bc]c来匹配字符串aac、abc、acc、adc时，则只有abc和acc两个字符串被成功匹配，而aac和adc则匹配不成功
^	表示匹配目标字符串的开始位置。例如，当用^ac正则表达式匹配字符串aac、abc、acc、adc时，则只有字符串acc被成功匹配。虽然字符串aac中也包含子字符串ac，但是该字符串中出现ac字符串的位置，并不是在开始
[^]	表示匹配目标字符串中，不包括被中括号括起来的字符集。例如，当使用正则表达式a[^bd]来匹配字符串aac、abc、acc和adc时，则只有字符串aac和acc可以匹配成功。因为其他两个字符串中包含了b或d字符
\$	表示从目标字符串的结束位置开始匹配。例如，当用ac\$正则表达式匹配字符串aac、abc、acc、adc时，则只有字符串aac会被成功匹配。虽然字符串acc中也包含字符串ac，但是该字符串中出现ac字符串的位置并不是在该字符串的结尾
() 小括号	用于指定一个子正则表达式。例如，当用([abc])-(ghi)正则表达式匹配字符串ab-gh、ac-km、ac-hi、cd-hg时，则可以成功匹配出ab-gh和ac-hi字符串。而且，在这种情况下，如果不使用子正则表达式，则对应的正则表达式将不太好看
* 星号	用于指定匹配该元字符前面子正则表达式零次或多次。例如，可以用ab*正则表达式匹配以ab开头的任意字符串
{n,m}	m和n均为非负整数，其中n<=m。最少匹配n次且最多匹配m次。例如，字符串aac、abc、acc、adc，如果使用正则表达式a{2,3}进行匹配，则只有aac字符串可以匹配成功，因为只有aac字符串中至少包含了2个字符“a”
[a-z]	指定用于匹配的字符集范围。例如，[a-z]正则表达式可以匹配“a”到“z”范围内的任意小写的英文字符。而[0-9]则可以匹配0到9的任意数字。例如，用a[a-c]正则表达式匹配字符串aac、abc、acc、adc，则只有aac、abc和acc字符串能够成功匹配，而adc字符串则匹配不成功
	表示两个匹配条件做逻辑“或”运算。例如，当用ab ac正则表达式匹配字符串aac、abc、acc、adc时，则只有aac、abc、acc字符串能够成功匹配，而adc字符串则匹配不成功
\	转义符。前面已经介绍过了，在正则表达式中是会包含元字符的。而元字符本身也是一个普通字符，如本表中介绍的这些字符。这样，当在一个字符串中，这些作为元字符使用的字符本身也是被匹配的对象时，则需要使用转义符对它们进行转义，否则系统将无法识别出它们是作为普通字符使用，还是被当作元字符使用，比如，\\(\\)则表示正则表达中的普通小括号

以上对正则表达式的知识做了一些简单介绍。但是，有关正则表达式的知识内容却远远不止以上介绍的这些。要完整地介绍正则表达式的知识，则需要另外写一本很厚的专门书籍。而且，书写正则表达式会涉及很多技巧，往往一个专业人士写出的一个稍微复杂一点的正则表达式，就会让不那么专业的人士觉得非常晦涩难懂。不过幸运的是，在我们日常管理和维护监控系统的过程中，一般很少书写复杂的正则表达式。所以，通过上面的介绍，结合接下来要介绍

的配置低级自动发现监控项目的过程中，所涉及到的正则表达式的知识应该能满足我们日常管理和维护 Zabbix 系统的需要。

5.3.2 正则表达式配置

上一节对正则表达式的相关知识做了简单介绍，接下来介绍如何在 Zabbix 中配置正则表达式，以及如何使用所配置的正则表达式。

依次选择菜单项“高级配置”→“常规”，并从“配置图形界面”页面的右边下拉菜单里选择“正则表达式”菜单项，系统将打开“配置正则表达式”页面，该页面显示了当前系统中，已经配置的正则表达式的信息列表。在这个列表页面上，单击页面右边的“新建正则表达式”按钮，系统将打开“配置正则表达式”表单页面，如图 5-7 所示。

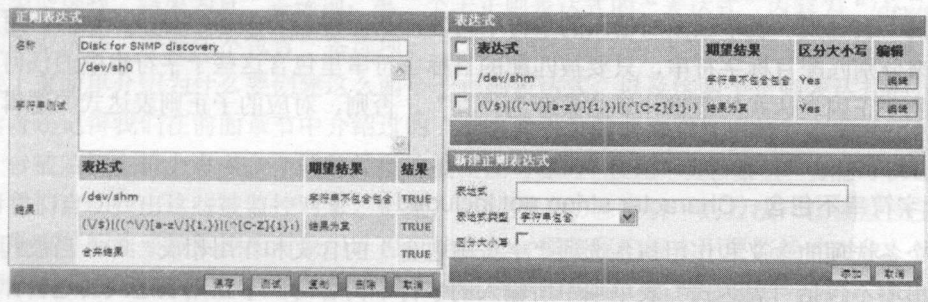


图 5-7 配置正则表达式表单页面截图

从图 5-7 中可以看出，“配置正则表达”表单页面上的内容主要由两部分组成。在页面左边的表单表上，可以输入新创建正则表达式的名称，且可以输入字符串对正则表达式的计算结果进行验证；而页面右边的表单用于创建当前正则表达式的子正则表达式。在 Zabbix 系统中，一个正则表达式可以由多个子正则表达式组成，而所有子正则表达式经过逻辑与运算后，所得的结果将作为其父正则表达式的计算结果。

接下来创建一个当使用低级自动发现功能去发现被监控主机上磁盘分区信息时，所需要使用的正则表达式“Disk for SNMP discovery”。如图 5-7 所示，首先在“配置正则表达式”表单页面左边的“名称”表单项里填写“Disk for SNMP discovery”，这个即为我们所创建的正则表达式的名称，在将来要使用这个正则表达式时，就是使用@Disk for SNMP discovery 形式来引用这个正则表达式。即，引用一个正则表达式时，使用@+正则表达式名称的格式。正则表达式名称可以由任何 Unicode 编码的字符组成的字符串，包括逗号和空格。但是，一般情况下，建议在正则表达式名称中尽量不要使用逗号，以免在引用正则表达式时出现意想不到的错误。

单击“配置正则表达式”表单页面上右边的“新建”按钮，系统将在当前页面上，增加一个用于添加子正则表达式项的“新建正则表达式”表单域。通过这个表单域，就可以在当前正则表达式里添加子正则表达式。其中，“表达式”表单项即是我们输入子正则表达式的地方。而“表达式类型”下拉菜单则用于选择要创建的子正则表达式的类型，它有 5 个备选项，这 5 个备选项的含义和作用如下所述。

1. 字符串包含(Character string included)

如果选择这个备选项则表示，当被匹配的目标字符串中，包含当前子正则表达式中字符串

内容时，也即在“新建正则表达式”表单域中“表达式”表单项里所输入的字符串内容，那么该子正则表达式的逻辑计算结果为“真”。

2. 字符串包含任何字符 (Any character string included)

这个备选项如果仅从字面的含义来理解，则很让人费解，而且很容易产生歧义。那么，这个备选项究竟是什么意思呢？我们举个例子来说明，可能就很容易理解了。假如我们在“表达式”表单项里输入“aaa/bbb/ccd/ddd”字符串，然后分隔符选择“/”，那么，只要被匹配的目标字符串中包含“aaa”、“bbb”、“ccd”、“ddd”等字符串中的任何一个，则该子正则表达式的逻辑计算结果即为“真”。否则，如果被匹配的目标字符串中，不包含上述4个字符串中的任何一个，则该子正则表达式的逻辑计算结果就为“否”。通过上面的例子，你大概已经明白了这个备选项的含义和作用了吧。即，如果选择这个备选项，那么系统将使用指定的分隔符来切割“表达式”表单项里所输入的字符串内容，从而得到若干子字符串。然后，用这些子字符串来分别匹配目标字符串，只要被匹配的目标字符串里包含这些子字符串中的任何一个，则对应的子正则表达式的逻辑计算结果即为“真”。否则，对应的子正则表达式的逻辑计算结果为“否”。

3. 字符串不包含 (Character string not included)

这个备选项的含义和作用与备选项“字符串包含”的含义和作用相反。即，当被匹配的目标字符串中不包括“表达式”表单项中所输入的字符串内容时，子正则表达式的逻辑计算结果为“真”。否则，子正则表达式的逻辑计算结果为“否”。

4. 结果为真 (Results is TRUE)

这个备选项的含义是，当子正则表达式在被匹配的目标字符串中匹配成功时，则子正则表达式的逻辑计算结果为“真”，否则，该子正则表达式的逻辑计算结果为“否”。

或许，你有一些迷惑了：正则表达式的主要作用本来就是用来匹配字符串的，至少在 Zabbix 监控系统中是这样的。那么，既然已经有了前面的三个备选项“字符串包含”、“字符串包含任何字符”和“字符串不包含”，那还要这个所谓的“结果为真”的备选项做什么用呢？它的作用和含义不是跟“字符串包含”备选项的作用和含义是一样的吗？是这样的，这个备选项跟我们在前面介绍的“字符串包含”、“字符串包含任何字符”和“字符串不包含”这3个备选项在含义和作用上还是有很大区别的。当选择“字符串包含”、“字符串包含任何字符”和“字符串不包含”3个备选项中的一个时，我们在“表达式”表单项里所输入的内容将被系统视作普通的字符串看待，而不是将其视作真正意义上的正则表达式。所以，此时系统在匹配时，是拿我们在“表达式”表单项里所输入的内容在目标字符串中查找，只有找到了完全一致的内容，才被视作是匹配成功。然而，选择“结果为真”这个备选项则不同，它所对应的匹配是真正意义上的正则表达式模式的匹配。

同样，也来个简单的例子加以说明，这样，读者可能一看就明白了它们之间的差别了。例如，我们在“表达式”表单项里输入“[abc]”这样一个字符串，然后分别选择“字符串包含”和“结果为真”这两个备选项进行测试。当选择“字符串包含”这个备选项时，被匹配的目标字符串中必须要完整地包含“[abc]”这个字符串，子正则表达式的逻辑计算结果才会是“真”。然而，当选择“结果为真”这个备选项时，则只要被匹配的目标字符串中包含“a”、“b”、“c”三个字符中的任何一个，相应的子正则表达式的逻辑计算结果即为“真”。

5. 结果与否(Results is FALSE)

这个备选项的含义和作用正好与“结果为真”这个备选项的含义和作用相反，即当所输入的正则表达式，在被匹配的目标字符串中没有被成功地匹配到，则该子正则表达式的逻辑计算结果为“真”，否则其逻辑计算结果为“否”。类似的，该备选项和“字符串不包含”备选项的区别与“结果为真”和“字符串包含”备选项的区别是一样的，即，当选择该备选项时，所对应的子正则表达式才是真正意义上的正则表达式。而选择“字符串不包含”备选项时，系统实际所作的操作只是在目标字符串中查找指定的字符串。

好了，介绍完“表达式类型”各个备选项的含义和作用后，接下来我们来完成前面没有完成的“Disk for SNMP discovery”正则表达式的配置。为完成这个正则表达式的配置，需要创建两个子正则表达式。第一个子正则表达式的“表达式”内容为“(\\$(^(\^[a-z\|]{1,})|(\^[C-Z]{1,}):)”，表达式类型选择“结果为真”备选项；第二个子正则表达式的“表达式”内容为“/dev/shm”，表达式类型选择“字符串不包含”备选项。

接下来简单介绍为什么要创建这么两个子正则表达式，以及这两个正则表达式的含义是什么。读者还记得我们在前面章节中介绍过的“动态索引”的概念和作用吗？前面介绍过，当不确定一台被监控主机上有多少个同种类型的被监控对象时，使用“动态索引”功能可以让 Zabbix 系统帮我们自动找出这些被监控对象。但是，当使用“动态索引”的方法，在被监控主机上所找出的所有可能需要被监控的对象中，也可能会包含那些我们根本不要监控的对象。例如，当通过动态索引的方法找出一台被监控主机上所有网卡信息时，系统会将回环网卡(lo)也找出来。但是很显然，一般来说不需要对回环网卡进行任何监控。所以，需要将这些不需要监控的对象给排除掉，或者说只选择需要监控的对象进行监控。再比如，当使用下列命令查看一台被监控主机上磁盘分区的信息时，系统将会显示出如图 5-8 所示的信息。

```
shell> snmpwalk -v2c -c snmp@iwwgame.com 10.1.1.158 HOST-RESOURCES-MIB::
hrStorageDescr
```

```
HOST-RESOURCES-MIB::hrStorageDescr.1 = STRING: Physical memory
HOST-RESOURCES-MIB::hrStorageDescr.3 = STRING: Virtual memory
HOST-RESOURCES-MIB::hrStorageDescr.6 = STRING: Memory buffers
HOST-RESOURCES-MIB::hrStorageDescr.7 = STRING: Cached memory
HOST-RESOURCES-MIB::hrStorageDescr.10 = STRING: Swap space
HOST-RESOURCES-MIB::hrStorageDescr.31 = STRING: /
HOST-RESOURCES-MIB::hrStorageDescr.35 = STRING: /dev/shm
HOST-RESOURCES-MIB::hrStorageDescr.36 = STRING: /boot
HOST-RESOURCES-MIB::hrStorageDescr.39 = STRING: /backup
```

图 5-8 磁盘分区信息

从图 5-8 可以看出，当查询被监控主机上磁盘分区信息时，SNMP 服务会将主机上物理内存、虚拟内存、内存缓存与缓冲区以及交换分区的信息都给我们列出来。很显然，这些与内存相关的信息，并不是我们想监控的磁盘分区信息范畴。因此，当需要监控主机的磁盘分区信息时，就需要使用正则表达式，将这些不需要的内容给排除掉。

前面所创建的正则表达式“Disk for SNMP discovery”中的两个子正则表达式，其作用就是从如图 5-8 所示的返回信息中，过滤出所需要的磁盘分区信息。在下一节我们将要介绍的磁盘分区低级自动发现功能中，Zabbix 系统首先会执行类似于上面执行的 snmpwalk 命令，从被监控主机上获取到如图 5-8 所示的返回信息，从而得到被监控主机上各磁盘分区，包括系统内存的描述信息字符串：“Physical memory”、“Virtual memory”、“Memory buffers”、“/”、“/boot”、“/backup”等。然后，Zabbix 系统执行在前面所定义的正则表达式“Disk for SNMP discovery”，从而过滤出所要监控的磁盘分区描述信息：“/”、“/boot”和“/backup”。

接下来分析上述第一个子正则表达式“(\\$(^(\^[a-z\|]{1,})|(\^[C-Z]{1,}):)”的含义。这个子

正则表达式由两个“或”运算符 (|) 连接三个子正则表达式组成。其中，子正则表达式“(V\$)”的含义是，从目标字符串中匹配出以“/”字符结尾的字符串。在系统所查询到的所有磁盘分区描述信息当中，只有根分区描述信息符合这个子正则表达式的规则。因此，这个子正则表达式会将根分区的描述信息给匹配出来。第二个子正则表达式“((^V)[a-z]{1,})”能匹配出以“/”开头，并且包含至少一个 a 到 z 字母的字符串。这样，通过第二个子正则表达式可以将“/boot”、“/backup”、“/dev/shm”等磁盘分区描述信息给匹配出来。而第三个子正则表达式“(^[C-Z]{1,}):”，则用于匹配被监控主机操作系统是 Windows 操作系统的磁盘分区描述信息。它的含义是：匹配出目标字符串中以字母 C 到 Z 中任何一个大写字母开头，且包括冒号(:)的字符串。这样，通过这个子正则表达式，就可以将类似于“C:\Label: Serial Number bc4dfbfe”、“D:\Label:Serial Number fe44839b”、“E:”等 Windows 系统下的磁盘分区描述信息给匹配出来了。

我们知道，在很多 Linux 发行版中，系统默认会开启临时文件系统 (tmpfs)。这个时候，在系统中会有一个虚拟的磁盘分区描述信息/dev/shm。一般情况下，当监控一个系统的磁盘分区信息时，不需要对这个临时文件系统进行监控。所以，需要在正则表达式里，将这个虚拟磁盘分区的描述信息给排除掉。在“Disk for SNMP discovery”正则表达式中所创建的第二个子正则表达式，就是用于排除这个虚拟磁盘分区信息的。因为，临时文件系统所对应的虚拟磁盘分区信息是相对固定的，所以，在正则表达式“Disk for SNMP discovery”中的第二个子正则表达式中，只需将包含字符串“/dev/shm”的字符串，从目标字符串中排除掉就可以了。

综上所述，当 Zabbix 系统根据我们所作的配置，从被监控主机上读取所有磁盘分区信息（包括虚拟磁盘分区信息）后，执行正则表达式“Disk for SNMP discovery”，对所取出的磁盘分区信息进行匹配和过滤。匹配和过滤后的磁盘分区信息，即为我们需要监控的磁盘分区信息。

5.3.3 低级自动发现功能

通过低级自动发现 (Low-level discovery, LLD) 功能，可以让 Zabbix 为我们在不同的被监控主机上，自动创建监控项目、触发器、数据图和图表等。当然，使用低级自动发现功能时，低级自动发现规则还是需要我们定义的。但是，可以将低级自动发现规则定义在一个主机模板上，然后将这个模板关联到不同的主机上，则这些关联了该模板的主机上，就会自动地被添加上所定义的监控项目、触发器及数据图等。例如，我们在前面章节中所提到的，不同的被监控主机上可能有不同数量的网卡和磁盘分区。这个时候，就可以使用低级自动发现功能，让 Zabbix 系统自动为我们查找，被监控主机上需要被监控的对象的数量，并自动地在被监控主机上添加上发现的监控项目。特别是在对网络交换机的网络端口进行监控时，低级自动发现功能就显得非常有用。假如有一台 48 口的网络交换机，需要对其每个网络端口的进出流量进行监控。此时，如果不使用低级自动发现功能，则需要手工为这台交换机的每个网络端口，分别配置流入流量监控项目和流出流量监控项目以及每个网络端口的进出流量总计监控项目，最后还要为每个网络端口创建数据图。这样，48 个端口的进出流量就 96 个监控项目，再加上在每个网络端口上要对其进出流量求和，这样每个网络端口又要再增加一个监控项目，同时，还需要为每个网络端口配置数据图等。最后，对于这台 48 口的网络交换机，我们总共需要配置 144 个监控项目和 48 个数据图。这么多的监控项目 and 数据图，如果完全依靠手工来添加，其工作量将是非常巨大的，而且还非常容易出错。但是，如果使用低级自动发现功能，那么只需配置一个进流量监控项目、一个出流量监控项目、一个进出流量求和的监控项目以及一个数据图就可以了。也就是说，总共只需配置三个监控项目和一个数据图，这样工作量会大大减小。

在 Zabbix 系统中实现了三种低级自动发现机制，它们分别是通过被监控设备代理组件 (zabbix agent) 所实现的低级自动发现功能、通过 SNMP 协议采集数据方式实现的低级自动发现功能和用户自定义的低级自动发现功能。这三种低级自动发现机制的不同，主要体现在数据采集方式和系统预定义的宏变量上。当使用被监控设备代理组件实现低级自动发现功能时，自动发现规则类型需要选择 zabbix agent。此时，系统已经为我们预定义了 {#FSNAME}、{#FSTYPE}、{#IFNAME} 等宏变量。而当通过 SNMP 协议采集数据方式实现低级自动发现功能时，自动发现规则类型需要选择 SNMPv1、SNMPv2 或 SNMPv3 中的一种，此时，系统已经为我们预定义了 {#SNMPINDEX} 和 {#SNMPVALUE} 两个宏变量。而当使用用户自定义的低级自动发现功能时，则自动发现规则类型可以由我们自己选择，而此时在自动发现规则执行过程中所需要的宏变量，需要由我们通过脚本或其他方式定义。但是，不管上述的这些宏变量是由系统预定义的，还是我们自己定义的，都可以在以下这些地方来使用它们。

1. 对于监控项目样板而言，可以将上述这些宏变量使用在以下这些地方：
 - 监控项目样板的名称中。
 - 监控项目样板的关键字中。
 - 监控项目样板的 OID 值中。
 - 通过计算采集监控数据所对应的监控项目样板的计算表达式中。
 - 通过 SSH 协议和 Telnet 协议采集监控数据所对应的监控项目样板的脚本程序中。
 - 数据库监控类监控项目样板的参数中。
2. 对于触发器样板，可以将这些宏变量使用在以下这些地方：
 - 触发器样板的名称中。
 - 触发器样板的触发器表达式中。
3. 对于数据图样板，可以将这些宏变量使用在数据图样板的名称中。

5.3.4 配置磁盘分区监控项目

前面已经对 Zabbix 中的低级自动发现功能做了简单的介绍，并且已经创建了一个名为“Disk for SNMP discovery”的正则表达式。接下来介绍一下，如何通过 SNMP 协议采集数据的方式，实现自动发现被监控主机上的磁盘分区信息，并且使用我们之前所创建的正则表达式，过滤掉不需要监控的磁盘分区信息。最后，让 Zabbix 系统自动创建一批，针对过滤后所得到的磁盘分区的监控项目。这些监控项目用于采集被监控主机上对应磁盘分区空间使用情况的数据。依据这些监控项目，再创建一批触发器，以实现所监控的磁盘分区空间使用率达到一定的值后，相应的触发器被触发，进而实现当磁盘分区空间使用率达到一定的值后，向指定人员发送报警信息的功能。

1. 配置低级自动发现规则

依次选择“系统配置”→“模板”菜单项，在打开的“配置模板”列表页面上，找到之前所创建的 test1 模板所在的行，并单击该行上“自动发现”字段上的超链接。接下来，在“配置自动发现规则”页面上单击“新建自动发现规则”按钮，系统将打开配置“自动发现规则”的表单页面，如图 5-9 所示。

从图 5-9 可以看出，配置低级自动发现规则的表单页面内容和配置一个普通的监控项目的表单页面的内容差别不大。实际上，在 Zabbix 系统中，低级自动发现规则也是被当作监控项目

来看待的。只是这个监控项目在采集数据时，返回的信息是一个信息列表。而该信息列表中的每一行即为系统所发现的一个潜在的（因为可能会被正则表达式过滤掉，所以并不是每一个被发现的对象都会被监控）被监控对象，并且系统根据用户所配置的规则，自动为这些被发现的监控对象添加上监控项目。

在图 5-9 所示的表单页面上，大部分表单项的含义和作用都比较容易理解，相信读者看到这些表单项的名称就知道它们的含义和作用，在此也就不再一一赘述，而只对以下几个容易让人迷惑或不好理解的表单项的作用和含义，稍加说明和解释：

- ❑ **类型。**因为这里是准备通过 SNMP 协议采集数据的，所以当然需要使用与 SNMP 协议有关的数据采集方法了。所以，这里选择“SNMPv2 agent”类型，当然也可以选择“SNMP v1 agent”或“SNMPv3 agent”类型，只是配置方法稍有不同而已，并没有本质上的差别。
- ❑ **关键字(key)。**你可以按照我们这里所输入的内容进行输入，也可以自行定义一个关键字，只要你所定义的关键字符合 Zabbix 系统对监控项目关键字的命令规范和要求即可。

名称 发现系统磁盘分区

类型 SNMPv2 agent

Key hrStorageDescr 选择

SNMP OID HOST-RESOURCES-MIB::hrStorageDescr

SNMP community {\$SNMP_COMMUNITY}

端口

更新间隔(秒) 60

Flexible intervals

间隔	周期	动作
No flexible intervals defined.		

New flexible interval

间隔(秒)	50	周期	1-7,00:00-24:00	添加
-------	----	----	-----------------	----

Keep lost resources period (in days) 30

过滤 宏变量 {\$SNMPVALUE} 正则 @Disk for SNMP disco

描述 The rule will discover all dis partitions matching the global regexp "Storage devices for SNMP discovery".
{\$SNMP_COMMUNITY} is a global macro.

状态 启用

图 5-9 配置低级自动发现规则表单页面截图

- ❑ **SNMP OID。**这个表单项的内容不能随意修改，在这个表单项里只能输入 MIB 库中存在的 OID 值，否则对应的自动发现规则将无法正常工作。
- ❑ **SNMP community。**这个表单项用于配置，连接被监控主机上 SNMP 服务所需的团体名。在该表单项里所填写的团体名，要与被监控主机上 SNMP 服务的配置文件中所配置的团体名保持一致。这里填写的是一个全局宏变量，你也可以使用宏变量，但需要在系统中事先配置好所对应的宏变量。
- ❑ **更新间隔（秒）。**这个表单项用于配置，系统间隔多长时间执行一次自动发现规则，以获取被监控主机上的磁盘分区变化情况。一般来说，一台主机上磁盘分区不太会频繁变动。所以，这个表单项的值可以配置得大一点，以减轻对 Zabbix 服务器性能的压

力。但是，如果这个配置项的值被设置得特别大也会有一个缺点，那就是被监控主机上磁盘分区情况有变化时，不能很快地反映到监控项目上。特别是，当被监控主机上新增加了磁盘分区后，至少需要经过这个表单项所指定的时间间隔后，新的监控项目才会被自动地添加到监控系统中。

- ❑ **Keep lost resources period(in days)**。该表单项用于指定，当某台被监控主机之前通过该自动发现规则，发现了某个被监控对象，并为其添加了一个相对应的监控项目，后来，系统在运行同一个自动发现规则时，又没有再次发现该被监控对象时，之前被添加的监控项目在系统中所保留的天数。
- ❑ **宏变量**。该表单项用于指定被过滤内容的宏变量名称，即对应宏变量中存放的是，将被“正则”表单项中所指定的正则表达式过滤的目标字符串。对于通过被监控设备代理 zabbix agent 实现的低级自动发现和通过 SNMP 协议采集数据方式实现的低级自动发现，这里所输入的宏变量，只能是系统中预定义的 {#FSNAME}、{#FSTYPE}、{#IFNAME}、{#SNMPINDEX} 和 {#SNMPVALUE} 宏变量中的一个。而对于用户自定义的低级自动发现，则这个表单项里所输入的宏变量，必须要跟被监控主机返回给 Zabbix 系统的，被发现的潜在被监控对象信息列表中，所包括的宏变量保持一致。
- ❑ **正则**。“正则”表单项里指定用于过滤系统所发现的潜在监控对象的正则表达式。这里填写在前面所创建的正则表达式 @Disk for SNMP discovery。

完成上述表单项的配置后，单击页面上的“保存”按钮，将我们所配置的自动发现规则配置信息保存到系统。这样就完成了一个低级自动发现规则的配置。

2. 配置项目样板

虽然前面已经配置了一个“发现系统磁盘分区”的低级自动发现规则。但是，在低级自动发现功能中，如果仅有自动发现规则，系统仍然不会自动添加监控项目。因此，接下来，需要为这个自动发现规则定义项目样板。只有定义好了项目样板，Zabbix 系统才会根据项目样板中所配置的信息，自动为通过自动发现规则所发现的被监控对象，添加上相应的监控项目。

配置项目样板的方法也很简单，依次选择菜单项“系统配置”→“模板”，在系统打开的“配置模板”列表页面上，找到之前所创建的 test1 模板所在的行，并单击该行上“自动发现”字段上的超链接。并在系统打开的“自动发现规则”列表页面上，找到刚刚所创建的“发现系统磁盘分区”自动发现规则，同时单击该行上“项目样板”字段上的超链接。系统将打开“配置项目样板”页面，在该页面上单击“新建项目样板”按钮，系统将打开如图 5-10 所示的“配置项目样板”表单页面。

对比如图 5-10 所示的“配置项目样板”表单页面上的表单项内容，与图 5-6 所示的“配置监控项目”表单页面上的表单项内容，我们很容易就可以发现，这两个页面上的表单项内容相差不多。所以，配置一个项目样板和配置一个普通的监控项目是差不多的，只是需要注意以下几点：

名称

分配的磁盘簇大小\$1

类型

SNMPv2 agent

Key

hrStorageAllocationUnits[{#SNMPVALUE}]

选择

SNMP OID

HOST-RESOURCES-MIB::hrStorageAllocationUnits.{#SNMPINDEX}

SNMP community

{\$SNMP_COMMUNITY}

端口

信息类型

数值(无符号整型)

数据类型

十进制

单位

乘自定义值

☐

1

更新间隔(秒)

30

Flexible intervals

间隔	周期	动作
No flexible intervals defined.		

New flexible interval

间隔(秒)	50	周期	1-7,00:00-24:00	添加
-------	----	----	-----------------	----

保留历史数据(天)

7

保留趋势数据(天)

10

取值方式

原始数据

显示值

原始数据

显示数据映射

图 5-10 配置项目样板表单页面截图

名称。虽说不管是普通监控项目的名称还是项目样板的名称，其内容都是可以支持宏变量的。但是，与普通监控项目不同的是，普通监控项目名称中不一定非要使用宏变量，而项目样板名称中必须要使用宏变量。这是因为，普通监控项目名称都是我们手工输入的，对于同一台被监控主机，一般不会有重复的现象，即使有不小心配置重复了的时候，则在配置监控项目时，系统就会给我们相应的提示，不允许我们添加具有相同名称的监控项目。然而，对于项目样板却不同了。既然叫做项目样板，那么在同一台被监控主机上，系统就有可能会依据同一个项目样板，自动添加上若干个类似的真实监控项目。这个时候，如果项目样板名称中没有一个宏变量，则这些被自动添加的真实监控项目（假如能添加成功），将会具有相同的监控项目名称，而这在 Zabbix 系统中是不允许的。所以，在项目样板的名称中至少要有一个宏变量，且要求该宏变量的值会依据系统所自动添加的真实监控项目的不同而不同。这样，当系统在某台主机上自动添加真实监控项目时，相对于当前所添加的真实监控项目的实际值，替换项目样板名称中的宏变量，从而使得基于同一个项目样板所生成的不同的真实监控项目，具有不同的监控项目名称。

对于安装了 Linux 操作系统的服务器而言，一般很难通过 SNMP 服务直接获取到某个磁盘分区以字节为单位的空间大小，而只能获取到磁盘分区的磁盘块（或者称作簇）的大小。所以，要想通过 SNMP 协议采集到某个磁盘分区以字节为单位的空间大小数据，我们只能先获取这个分区的块数，以及系统分配给该分区的每个数据块的字节数，然后将所得到的数据块数乘以每个数据块的字节数，就能得到该分区以字节为单位的空间大小数据了。所以，在这里首先需要创建一个名称为“分配的磁盘簇大小\$1”的项目样板，以采集系统分配给每个分区的单个数据块以字节为单位的空间大小数据。

其中, 宏变量\$1 指代的内容是, 对应项目样板关键字第一个参数的内容, 这里它的内容其实就是{#SNMPVALUE}宏变量所指代的内容, 关于这一点我们在前面章节中已经作过详细的介绍。实际上, 在这里如果不使用宏变量\$1, 而直接使用宏变量{#SNMPVALUE}, 其效果也是一样的。而宏变量{#SNMPVALUE}所指代的内容, 是依据被自动发现对象的不同而不同的, 且该宏变量的内容, 是经过正则表达式 Disk for SNMP discovery 过滤过的。所以, 当系统依据该项目样板生成实际监控项目时, 其实际监控项目名称内容将会依据实际所发现对象名称的不同而不同, 这就避免了同一台主机上监控项目名称重复的问题。

- ❑ **关键字(Key)。**“关键字”表单项里的内容可以由我们自行定义, 只要定义的关键字符合 Zabbix 系统对关键字的命名规范和要求即可。但是, 和项目样板名称一样, 项目样板的关键字中也必须至少带一个宏变量作为它的参数, 以避免不同的实际监控项目的关键字重复。
- ❑ **SNMP OID。**我们知道, 在系统的 MIB 库中, 一个 OID 值对应一个对象。所以, 对于普通的通过 SNMP 协议采集监控数据的监控项目, 它们的 OID 值是固定的。但是, 我们这里所配置的是项目样板, 在系统依据项目样板创建实际监控项目时, 其所创建的实际监控项目数量会依据被监控主机的不同而动态变化。况且, 对于不同的监控项目, 其 OID 值肯定是不一样的。所以没有办法将项目样板的 OID 值指定为一个固定的值。这个时候, 就需要使用到动态索引的方法了。根据前面章节中所介绍的动态索引的相关知识, 我们知道, 对于同一类型的对象, 它们 OID 值的左边大部分是相同的, 而不同的只是通过点号(.) 隔开的 OID 值中的最后一节, 并且我们将之称为索引。在低级自动发现过程中, Zabbix 系统会将各个对象的索引值都找出来, 并且将所找到的索引值记录到宏变量{#SNMPINDEX}中。这样, 在系统自动生成实际监控项目时, 只要将对象 OID 值的不变部分, 与宏变量{#SNMPINDEX}所指代的值拼接在一起, 就是形成了所要监控对象的 OID 值了。所以在配置项目样板时, 其 SNMP OID 表单项里的内容就应该填写成某类被监控对象 OID 值的固定部分{#SNMPINDEX}的形式, 比如, 在这里就应该写成 HOST- RESOURCES-MIB::hrStorageAllocationUnits.{#SNMPINDEX}的形式。

“配置项目样板”表单页面上其他表单项的内容, 读者可以根据自己的实际需要进行填写。在完成所有表单项内容的填写后, 单击页面下部的“保存”按钮, 以完成项目样板的配置。

以上只是创建了一个项目样板, 以用于获取磁盘分区每个数据块被分配的字节数大小。如前文所述, 为了获取磁盘分区的空间使用率, 我们还需要获取分区被分配的总数据块数和已使用的数据块数, 并以此为基础, 计算出分区总的空间大小和已使用的空间大小。为此, 我们还需要创建如下这些项目样板。下面这些项目样板的创建过程与前面所创建的项目样板的创建过程是类似的。故此, 就不一一赘述它们的创建过程了, 请读者参考下面所给出的各字段的内容, 自行创建好这些项目样板。

项目样板二 (将前面刚刚所创建的项目样板记作项目样板一)

名称: 磁盘簇总计 \$1。

类型: SNMP v2 agent。

Key: hrStorageSize[{#SNMPVALUE}].

SNMP OID: HOST-RESOURCES-MIB::hrStorageSize.{#SNMPINDEX}。

信息类型：数值（无符号整型）。

项目样板三

名称：磁盘簇已使用 \$1。

类型：SNMP v2 agent。

Key: hrStorageUsed[{#SNMPVALUE}]]。

SNMP OID: HOST-RESOURCES-MIB::hrStorageUsed.{#SNMPINDEX}。

信息类型：数值（无符号整型）。

项目样板四

名称：磁盘空间总计 \$1。

类型：Calculated。

Key: hrStorageSizeInBytes[{#SNMPVALUE}]]。

公式：

last("hrStorageSize[{#SNMPVALUE}]]") * last("hrStorageAllocationUnits[{#SNMPVALUE}]]")

信息类型：数值（无符号整型）。

单位：B。

项目样板五

名称：磁盘空间已使用 \$1。

类型：Calculated。

Key: hrStorageUsedInBytes[{#SNMPVALUE}]]。

公式：

last("hrStorageUsed[{#SNMPVALUE}]]") * last("hrStorageAllocationUnits[{#SNMPVALUE}]]")。

信息类型：数值（无符号整型）。

项目样板六

名称：磁盘空间使用率 \$1。

类型：Calculated。

Key: hrStorageUsage[{#SNMPVALUE}]]。

公式：

100 * last("hrStorageUsed[{#SNMPVALUE}]]") / last("hrStorageSize[{#SNMPVALUE}]]")。

信息类型：数值（无符号整型）。

单位：%。

如果上述 6 个项目样板均创建完成，则它们的配置信息如图 5-11 所示。

配置向导	名称	触发器 Key	间隔	历史数据	趋势数据	类型	监控分类	状态	错误
<input type="checkbox"/>	Template SNMP Generic: Device uptime	sysUpTime	60	7	365	SNMPv2 agent	General	启用	<input checked="" type="checkbox"/>
<input type="checkbox"/>	Template SNMP Generic: Device name	sysName	3600	7		SNMPv2 agent	General	启用	<input checked="" type="checkbox"/>
<input type="checkbox"/>	Template SNMP Generic: Device location	sysLocation	3600	7		SNMPv2 agent	General	启用	<input checked="" type="checkbox"/>
<input type="checkbox"/>	Template SNMP Generic: Device description	sysDescr	3600	7		SNMPv2 agent	General	启用	<input checked="" type="checkbox"/>
<input type="checkbox"/>	Template SNMP Generic: Device contact details	sysContact	3600	7		SNMPv2 agent	General	启用	<input checked="" type="checkbox"/>

图 5-11 项目样板信息列表

3. 应用低级自动发现规则

当完成上述低级自动发现规则和项目样板的配置后，就可以将带有这个低级自动发现规则和项目样板的模板关联到被监控主机上。完成这个操作几分钟之后，依次选择“系统配置”→“主机”菜单项，并在“配置主机”列表页面上，找到关联了上述模板的主机所在的行，并单击该行“项目”字段上的超链接，就可以看到如图 5-12 所示的，系统依据所创建的低级自动发现规则和项目样板，自动在对应主机上所创建的实际监控项目列表信息。

配置向导	名称	触发器	Key	间隔	历史数据	趋势数据	类型	监控分类	状态	错误
	Disk partitions: 磁盘簇总计 /		hrStorageSize[/]	30	7	10	SNMPv2 agent	磁盘分区	启用	
	Disk partitions: 磁盘簇总计 /boot		hrStorageSize[/boot]	30	7	10	SNMPv2 agent	磁盘分区	启用	
	Disk partitions: 磁盘簇已使用 /	The item is not discovered anymore and will be deleted in 29d 23h 46m (on 30 九月 2014 at 08:22:27).								
	Disk partitions: 磁盘簇已使用 /boot		hrStorageUsed[/boot]	30	90	365	SNMPv2 agent	磁盘分区	禁用	
	Disk partitions: 磁盘空间使用率	触发器 (2)	disk.usage[/]	60	7	14	Calculated	磁盘分区	启用	
	Disk partitions: 磁盘空间使用率	触发器 (2)	disk.usage[/boot]	60	7	14	Calculated	磁盘分区	启用	
	Disk partitions: 分配的磁盘簇大小 /		hrStorageAllocationUnits[/]	30	7	10	SNMPv2 agent	磁盘分区	启用	
	Disk partitions: 分配的磁盘簇大小 /boot		hrStorageAllocationUnits[/boot]	30	7	10	SNMPv2 agent	磁盘分区	启用	
	Disk partitions: 分区总的磁盘空间 /		hrStorageSizeInBytes[/]	60	7	14	Calculated	磁盘分区	启用	

图 5-12 低级自动发现的监控项目列表

从图 5-12 可以看出，Zabbix 系统依据我们所配置的低级自动发现规则，在被监控主机上所创建的监控项目列表信息。这个列表上的监控项目，在被监控主机监控项目列表页面上显示时，系统会自动在监控项目名称前面，添加上系统创建它们所依据的低级自动发现规则名称。单击低级自动发现规则名称上的超链接，系统会打开对应低级自动发现规则上所配置的项目样板列表页面。

当某个之前通过低级自动发现规则所发现的实际监控项目，因为某种原因，在后来的自动发现过程中又丢失了（也就是系统没有再次发现对应监控项目）时，则系统在监控项目列表页面上显示该监控项目时，会在“错误”列，以一个橙色感叹号图标将该监控项目标识出来，如图 5-12 中“磁盘簇总计/boot”所在的行所示。而当用鼠标指向这个橙色感叹号图标时，系统会给出对应监控项目将在多长时间之后被删除的提示信息。而这个删除动作，是由 Zabbix 系统的相关进程根据我们所作的配置自动完成的。换句话说，在被监控主机层面上，我们是无法直接删除通过低级自动发现功能自动生成的监控项目的。如果要删除某个通过低级自动发现功能自动生成的监控项目，则我们要么修改正则表达式的过滤规则，要么在低级自动发现规则层面上删除生成相应监控项目的项目样板。但是，如果在低级自动发现规则层面上删除了某个项目样板，则依据该项目样板所生成的所有实际监控项目都将会被自动删除。

与不能在被监控主机层面上删除系统通过低级自动发现功能所创建的实际监控项目不一样，这类监控项目既可以在被监控主机层面上被禁用，也可以在自动发现规则层面上被禁用。当在自动发现规则层面上禁用了某个项目样板时，则系统依据该项目样板在被监控主机上所能创建的实际监控项目，仍然还会被自动创建，只是所创建的实际监控项目的状态也是禁用状态。

上面所述的，系统依据低级自动发现规则，在被监控主机上所创建的实际监控项目的显示规则、删除规则和禁用规则等，同样适用于我们在后续章节中将要介绍的，系统依据“触发器样板”和“数据图样板”，在被监控主机上自动创建的实际触发器和实际数据图。

5.3.5 配置网卡流量监控项目

在前一节，我们实际动手配置了一个低级自动发现规则和 6 个项目样板，这个低级自动发现规则，其数据是通过 SNMP 协议来采集的。本节再来配置一个低级自动发现规则，这个低级自动发现规则是通过被监控设备代理（Zabbix agent）来实现数据采集的。通过这个自动发现规则，Zabbix 系统将会发现被监控主机上所有网卡信息，并按照我们所配置的过滤条件，提取出需要监控的网卡信息。之后，依据这些被提取的需要被监控的网卡信息，Zabbix 系统自动在系统中添加上网卡进出流量监控项目。

既然是通过被监控设备代理组件，实现低级自动发现规则所需数据的采集，那么很显然，如果需要使用这个低级自动发现规则，则就需要在被监控主机上安装和配置好 Zabbix 系统的被监控设备代理组件，并启动它。这一部分工作，请读者在具体配置低级自动发现规则和项目样板之前，参考本书前面相关章节的介绍，自行完成。

1. 配置正则表达式

与前面配置磁盘分区监控项目类似，在配置自动发现规则之前，需要先配置一个正则表达式，以用于过滤被系统发现的网卡信息。这是因为，在一个系统中，往往除了配置有多张真实网卡之外，还可能因为业务的需要而配置了多张虚拟网卡。比如，操作系统在安装时会自动创建的回环网卡、宿主机上由虚拟机软件自动创建的多个虚拟网卡等。一般来说，对于这些虚拟网卡的状态和流量，不需要进行监控。所以，需要创建一个正则表达式，过滤掉这些虚拟网卡的信息。

接下来需要在系统中配置一个新的正则表达式。配置方法与前面配置正则表达式 Disk for SNMP discovery 的方法类似，表单中所需要填写内容如下：

名称。Network interfaces for zabbix agent discovery。

子正则表达式。(^eth[0-9]{1}\$)(^em[0-9]{1}\$)。

表达式类型。结果为真。

配置完成后的正则表达式内容如图 5-13 所示。

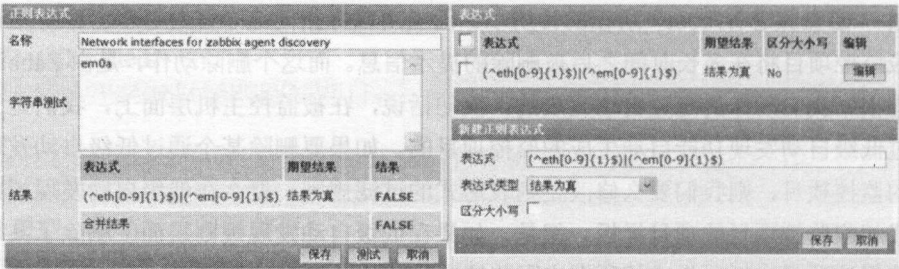


图 5-13 配置正则表达式表单页面截图

上述子正则表达式的含义是：提取出以 eth 开头且紧接着一个 0~9 数字的字符串，或者是以 em 开头紧接着一个 0~9 数字的字符串。因此，经过该子正则表达式过滤后的字符串内容，只能是形如 eth0、eth1、em0、em1 等形式的网卡名称。而像 eth10、eth12、lo、vnet0、vnet1 等形式的网卡名称，统统将会被过滤掉。

2. 配置自动发现规则

完成正则表达式的配置后，接下来，就可以按照前面配置低级自动发现磁盘分区的自动发

现规则的方法和流程，来配置自动发现网卡的自动发现规则。其中，配置表单页面上的部分表单项内容如下：

- 名称。发现系统网卡。
- 类型。Zabbix agent。
- Key。net.if.discovery。
- 宏变量。{#IFNAME}。
- 正则。@Network interfaces for zabbix agent discovery。

3. 配置项目样板

同样，在完成低级自动发现规则的配置后，也需要为所配置的低级自动发现规则配置项目样板。其中配置项目样板的方法，与在前面配置低级自动发现系统磁盘分区的项目样板的方法是类似的，在此就不再一一介绍每个项目样板的配置过程，仅列出配置这些项目样板的表单项时，部分表单项里所需要选填的内容，请读者自行完成这些项目样板的配置。

项目样板一

- 名称：{#IFNAME}网卡数据流出量。
- 类型：Zabbix agent。

Key: net.if.out[{#IFNAME},bytes]。

- 信息类型：数值（无符号整型）。
- 单位：bps。

项目样板二

- 名称：{#IFNAME}网卡数据流入量。
- 类型：Zabbix agent。

Key: net.if.in[{#IFNAME},bytes]。

- 信息类型：数值（无符号整型）。
- 单位：bps。

项目样板三

- 名称：{#IFNAME}网卡总流量。
- 类型：Calculated。

Key: net.if.total[{#IFNAME},bytes]。

公式：last("net.if.in[{#IFNAME},bytes]") + last("net.if.out[{#IFNAME},bytes]")

- 信息类型：数值（无符号整型）。
- 单位：bps。

如果上述三个项目样板均创建完成，则它们的配置信息如图 5-14 所示。

<input type="checkbox"/> 名称 *	Key	间隔	历史数据	趋势数据	类型	状态	监控分类
<input type="checkbox"/> {#IFNAME}网卡总流量	net.if.total[{#IFNAME},bytes]	30	90	365	Calculated	启用	网卡流量
<input type="checkbox"/> {#IFNAME}网卡数据流入量	net.if.in[{#IFNAME},bytes]	30	90	365	Zabbix agent	启用	网卡流量
<input type="checkbox"/> {#IFNAME}网卡数据流出量	net.if.out[{#IFNAME},bytes]	30	90	365	Zabbix agent	启用	网卡流量

图 5-14 项目样板信息列表

5.3.6 配置网络端口连接数监控项目

前面已经配置了两个低级自动发现规则，它们的自动发现机制分别是，使用通过 SNMP 协议采集数据方式实现低级自动发现和通过被监控设备代理组件采集数据方式实现低级自动发现。接下来，再来配置一个低级自动发现规则，这里我们通过自己编写脚本程序采集数据的方式来实现低级自动发现。

将要创建的这个低级自动发现的功能是，Zabbix 系统通过 SSH 协议采集数据的方式，连接到被监控主机上并执行该主机上的一个指定的脚本程序（`ldd_port_connects.sh`）。而该脚本程序在被成功执行后，将能返回被监控主机上所有被侦听的端口信息列表。然后，Zabbix 系统捕获这个端口列表信息，并根据项目样板的配置，执行被监控主机上的另一个脚本程序（仍然是通过 SSH 协议的方法）。而该脚本程序的功能是，返回被监控主机上指定端口连接状态是“ESTABLISHED”状态的连接数。

1. 实现原理及接口规范

虽然，如上所述，即将创建的自动发现规则是通过 SSH 协议采集数据的方法来获取被监控对象信息的。但是，实际上，在 Zabbix 系统中，可以基于任何一种监控数据采集方法来实现低级自动发现功能。那么，它们的实现原理和工作流程是怎么样的呢？

在前面，已经在系统中配置了两个低级自动发现规则及其项目样板。在配置这两个低级自动发现规则的过程中，我们发现，配置一个自动发现规则与配置一个普通的监控项目没有太大差别。所以，所谓自动发现规则，对于 Zabbix 系统来说，其实只是一个特殊的监控项目而已。当 Zabbix 服务器端执行一个自动发现规则时，它会像针对普通监控项目采集监控数据那样，通过指定的方法从被监控主机上采集所需要的数据。但是，不同的是，对于普通监控项目，被监控主机所返回的监控数据可能是单条的数据；而在系统执行一个低级自动发现规则时，被监控主机所返回的数据是 JSON 格式的宏变量到数值的数据列表。而每对这样的宏变量到数值的记录，对于 Zabbix 系统来说，就表示是系统所发现的一个潜在的（因为可能会被正则表达式过滤掉）监控对象，例如，图 5-15 所示即为在运行一个低级自动发现脚本程序时，被监控主机返回给 Zabbix 服务器端的信息列表。

```
root@localhost ~# ./ldd_port_connects.sh
{
  "data": [
    {"#PORTNUM": "10050"},
    {"#PORTNUM": "193"},
    {"#PORTNUM": "2208"},
    {"#PORTNUM": "3306"},
    {"#PORTNUM": "38825"},
    {"#PORTNUM": "5801"},
    {"#PORTNUM": "5901"},
    {"#PORTNUM": "6001"},
    {"#PORTNUM": "631"},
    {"#PORTNUM": "9000"} ]
}
```

图 5-15 被监控主机返回的自动发现信息列表

当 Zabbix 服务器端从被监控主机那里接收到如图 5-15 所示的自动发现信息列表后，它会每对宏变量到值的数据对，视为一个潜在监控对象，然后使用系统中所配置的正则表达式对接收到的信息进行过滤。最后，系统依据过滤后所得到的监控对象信息，并依据项目样板的配置，在指定主机上创建相应的监控项目。其中，自动发现信息列表中所包含的宏变量，如图 5-15 中所示的“`{#PORTNUM}`”，即为可以被使用到项目样板名称中或关键字参数的宏变量。而每个宏变量所对应的值，即为指定正则表达式所需要过滤的目标字符串。

经过上面的介绍已经不难理解，其实，在 Zabbix 系统中配置低级自动发现规则时，可以使

用 Zabbix 系统所支持的任何一种监控数据采集方法。只需 Zabbix 系统在执行所配置的低级自动发现规则时，相应的监控数据采集方法能返回一定格式的数据即可。而这种数据格式，即接口规范，为下列形式的 JSON 格式数据：

```
1. {
2.   "data": [
3.     { "#MACRO": "VALUE" },
4.     { "#MACRO": "VALUE" },
5.     { "#MACRO": "VALUE" },
6.     { "#MACRO": "VALUE" },
7.   ]
8. }
```

2. 编写数据采集脚本

接下来，为实现自动发现被监控主机上所有服务所侦听的端口号，以及这些被发现的端口中连接状态为“ESTABLISHED”的连接数，需要编写两个脚本程序，它们分别是 ldd_port_connects.sh 脚本程序和 get_conn_num.sh 脚本程序。其中，ldd_port_connects.sh 脚本程序应用在自动发现规则上，用于发现被监控主机上所运行的所有服务侦听的端口号，并给 Zabbix 服务器端返回如图 5-15 所示的自动发现信息列表数据。而脚本程序 get_conn_num.sh 应用在项目样板和实际监控项目上，用于采集指定端口号的连接状态为“ESTABLISHED”的连接数。以上两个脚本程序的程序源代码如下所示：

脚本 ldd_port_connects.sh 的程序源代码：

```
1. #!/bin/bash
2. /bin/echo "{"
3. /bin/echo -e "\"data\":["
4. FIRST_LINE="1"
5. for port in ` /bin/netstat -ntpl | /bin/grep "tcp" | /bin/awk '{print $4}'
6. | /bin/awk -F ":" '{print $2}' | /bin/sort -u`
7. do
8.   conn_num=` /bin/netstat -an | /bin/grep $port | /bin/grep "ESTABLISHED"
9. | /usr/bin/wc -l`
10.  if [ $FIRST_LINE == "1" ]
11.  then
12.    /bin/echo -e -n "{\"#PORTNUM\": \"$port\"}"
13.    FIRST_LINE="0"
14.  else
15.    /bin/echo -e -n ", \n{\"#PORTNUM\": \"$port\"}"
16.  fi
17. done
18. /bin/echo -e " ] \n} \n"
```

脚本 get_conn_num.sh 的程序源代码：

```
1. #!/bin/bash
2. if [ $# -lt 1 ]
3. then
4.   /bin/echo "-1"
5.   exit
6. fi
7. conn_num=` /bin/netstat -an | /bin/grep $1 | /bin/grep "ESTABLISHED"
8. | /usr/bin/wc -l`
9. echo $conn_num
```


3. 配置自动发现规则

与配置两个低级自动发现规则的操作类似，在 Zabbix 系统的 Web 页面上依次选择“系统配置”→“模板”菜单项，并在打开的“配置模板”列表页面上，找到我们之前所创建的 test1 模板所在的行，并单击该行上“自动发现”字段上的超链接。接下来，在“配置自动发现规则”页面上单击“新建自动发现规则”按钮，系统将打开配置“自动发现规则”的表单页面，该页面上各表单项按以下内容进行填写：

名称。自动发现服务侦听端口。

类型。SSH agent。

Key。ssh.run[service_portnum,,22]。

Authentication method。公钥。

用户名。useraccount。

公钥文件。id_rsa.pub。

Private key file。id_rsa。

已执行脚本。/home/useradccount/shell/ldd_port_connects.sh。

上述表单项中，验证方法（Authentication method）也可以选择“密码”选项。如果选择了“密码”选项，则需要输入被监控主机上 useraccount 账号所对应的密码。如果验证方法选择“公钥”方式，则需要提前按照我们在前面章节所介绍的方法，配置好被监控主机与 Zabbix 服务器之间公钥/私钥认证。如果在生成公钥/私钥时，输入过 Key passphrase，则在填写这个表单时，也应该在 Key passphrase 表单项里填写相应的密码。

“自动发现规则”表单页面上的其他表单项，你可以根据自己的需要，填写适当的内容。完成各表单项内容的填写后，单击页面下部的“保存”按钮，完成低级自动发现规则的创建。

4. 配置项目样板

与前面所配置的另外两个低级自动发现功能一样，要想让刚刚所创建的低级自动发现规则起作用，需要在这个规则上配置项目样板。配置项目样板的操作方法，可参考前面介绍的配置“低级自动发现磁盘分区”项目样板的操作方法。以下是在配置项目样板时，部分表单项里所需要输入的内容：

名称。{#PORTNUM}端口连接数。

类型。SSH agent。

Key。ssh.run[{#PORTNUM},,22]。

Authentication method。公钥。

用户名。useraccount。

公钥文件。id_rsa.pub。

Private key file。id_rsa。

已执行脚本。/useraccount/shell/get_conn_num.sh {#PORTNUM}。

完成上述配置之后，并将 test1 模板关联到监控主机上，则 Zabbix 系统就会自动执行自动发现脚本程序，以获取被监控主机上监听的所有网络端口列表，并依据所获取的端口列表，依次在被监控主机上自动创建相对应的监控项目。最后，Zabbix 系统在被监控主机上周期性地执行/useraccount/shell/get_conn_num.sh 脚本程序，以采集相应端口连接状态是“ESTABLISHED”的网络连接数。

在配置上述这个低级自动发现功能时，没有使用正则表达式对所采集到的网络端口列表信息进行过滤。所以，只要是被监控主机上所侦听的端口，Zabbix 系统都会自动为它在对应主机上创建一个相应的监控项目，以采集相对应的网络连接数。当然，如果有需要，同样也可以在自动发现规则上添加上某个正则表达式，以过滤所发现的端口列表信息。

5.4 数据图及其配置

对于许多监控项目，例如网卡的网络流量、系统的内存使用情况、系统负载以及应用系统的在线人数，等等，我们不但希望当它们的数据达到或超过某个阈值（当然也有可能是低于某个阈值）时，系统能给我们报警，而且还希望能直观地看到这些监控项目在某个时段内的数据走势图。从而为我们能找出这些监控项目数据的变化规律，为系统性能的调优、系统架构改造甚至 IT 基础设施的重构提供数据支持。Zabbix 系统所提供的数据图功能，就完全能满足这些需求。

5.4.1 数据图

在 Zabbix 系统中，系统提供了简单数据图(Simple graphs)和自定义数据图(Custom graphs)两种形式的数据图。当某个监控项目在配置时，其信息类型被配置成数值型的（包括无符号的整型和浮点型），则系统会自动为这个监控项目创建相对应的数据图（需要通过“数据总览”或“最新数据”页面来查看）。这种被 Zabbix 系统自动创建的数据图即被称为简单数据图，它是基于系统中某个单独监控项目所采集的监控数据的。换句话说，在简单数据图上，系统只能一次显示一个被监控项目所采集数据的数据图。比如说，我们在上面介绍如何配置网卡流量项目样板时，分别配置了网卡接收流量和发送流量两个项目样板。这样，当系统基于我们所配置的这两个项目样板，在系统中为某台被监控主机自动创建实际监控项目时，它会针对从该被监控主机上所发现的每一块网卡，分别创建对应的接收流量监控项目和发送流量监控项目。此时，如果要查看这块网卡的流量图，则需要分别针对这块网卡的接收流量和发送流量查看它们的数据图，没有办法在同一张数据图上，同时查看到这块网卡的接收流量图和发送流量图。

因此，为了克服简单数据图的上述缺点，Zabbix 系统也为我们提供了创建自定义数据图的功能。在自定义数据图上，Zabbix 系统可以依据多个监控项目所采集的数据，在同一张数据图上，分别绘出这些监控项目所对应的数据图。这样，在同一张自定义的数据图上，就可以方便地同时查看多个监控项目的数据图了。这样，不但简化了查看数据图的操作步骤，而且也便于对不同监控项目的数据走势进行比较。但是，自定义数据图也有一个缺点，那就是系统不会自动生成它，需要我们自己事先在系统中创建好。当然，不但可以将自定义数据图配置在模板上，而且还可以像定义“项目样板”一样，在低级自动发现规则上定义“数据图样板”。所以，从这个角度上来说，在 Zabbix 系统中使用自定义数据图也是很方便的。

5.4.2 读懂简单数据图

简单数据图虽然名称中带“简单”两个字，但是，要读懂它可不是那么简单。我们先来看看图 5-16 和图 5-17 所示的简单数据图。

细心的读者可能已经发现了，在图 5-16 和图 5-17 上，它们所标识的图题是一样的，都叫“监控服务器(192.168.5.139) 负载平均值(5 分钟)”。难道这是同一台被监控主机上同一个监控项目的两张简单数据图？是的，这就是同一台被监控主机上同一个监控项目的两张简单数据图。或许你又要问了，但是，这两张数据图的显示效果却差异非常明显，难道就是因为两张数据图所显示的时间范围不同而引起的？

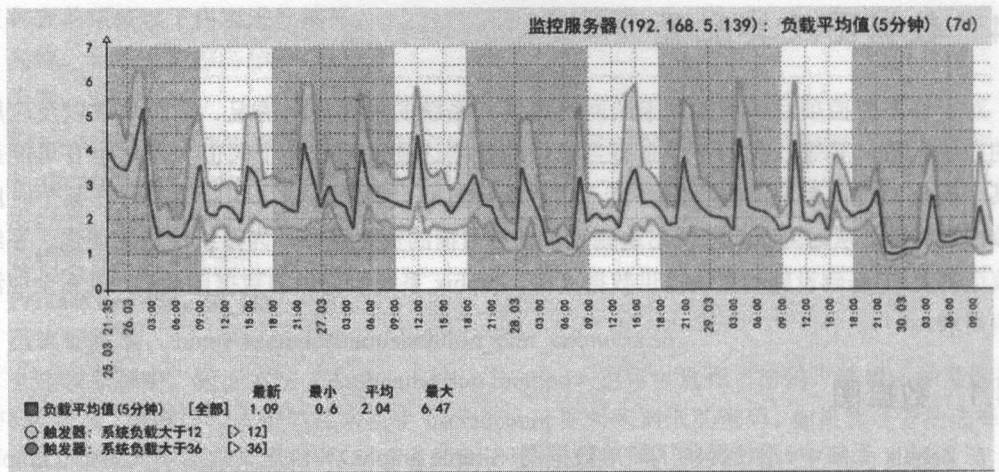


图 5-16 系统负载平均值数据图

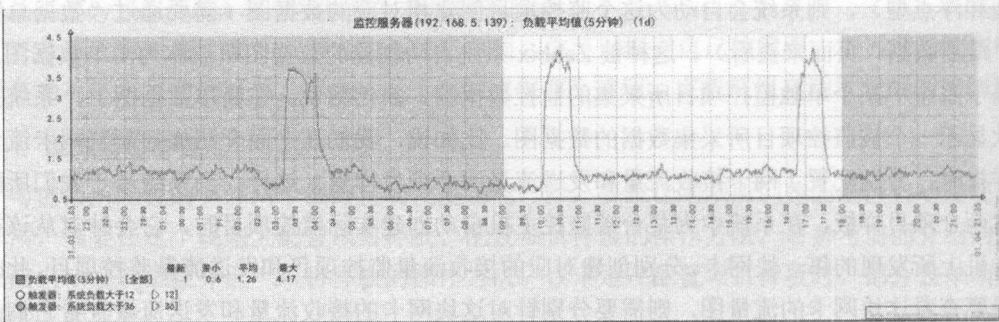


图 5-17 系统负载平均值数据图

你思考的一点都没有错，当 Zabbix 系统在显示一张简单数据图时，它会根据你要求显示的时间范围的不同，严格来说，是你要求显示的时间跨度的不同，而显示不同样式的数据图。当你要求显示的时间跨度比较小时，比如图 5-17 中选择的时间跨度为一天，则系统将会以一根单线来绘制这个数据图。此时，数据图上的曲线上的每一个点的值，系统都将尽量使用对应监控项目所采集的实际值。而当要求显示的时间跨度比较大时，比如图 5-16 中选择的时间跨度为 7 天，则系统将会在对对应数据图上分别用粉红色、深绿色和淡绿色绘出三条曲线，如图 5-16 所示的上、中、下三条曲线。那么，这三条曲线是怎么来的呢？当要求 Zabbix 系统为我们所绘制的数据图的时间跨度超过一定长度时，Zabbix 系统将不再直接用该数据图所对应监控项目所采集的实际数据作为绘图的数据。而是，首先将该项目在指定的时间范围内所采集的数据，按一小时为间隔，划分为等时间长度的数据集。然后，系统对所划分的每个数据集分别求最大值、平均值和最小值。最后，系统再用求出的这一系列最大值、最小值和平均值分别绘制曲线。这样，在数据图上就会呈现出三条曲线。它们分别是：用最大值绘制的曲线，毫无疑问就是数据图中

最上面的那条曲线；而用平均值和最小值绘出的曲线则分别对应于数据图上的中间那一条和最下面那一条曲线。而在最下面和最上面两条曲线之间，Zabbix 系统则用淡黄色来填充它。经过上述步骤的绘制，系统最后呈现给我们的数据图就类似于图 5-16 所示的样式。

在前面章节中介绍过，保存在 Zabbix 系统数据库中的监控项目所采集的监控数据分为“历史数据”和“趋势数据”这两种，而且这两种监控数据都是可以用来绘制数据图的。那么，我们怎么知道某个特定的数据图到底是基于“趋势数据”绘制的，还是基于“历史数据”绘制的呢？请读者仔细看图 5-16 和图 5-17 右下角用框框起来的位置。如果仔细看，在每张数据图的相同位置，系统都会写上类似“Date from trends. Generated in 0.17 sec”的说明。从这个说明中，就可以看出，系统在绘制这张数据图时所使用的数据来源，即到底是“历史数据”还是“趋势数据”。

那么，用户是否可以控制 Zabbix 系统在绘制指定的数据图时，是使用“历史数据”还是“趋势数据”呢？不可以，这个是由 Zabbix 系统主要依据以下两个因素，来决定到底使用哪种数据来绘制指定数据图。

- 监控项目上所配置的保留历史数据时间长度，也就是我们在配置监控项目时，通过“保留历史数据（天）”表单项所配置的数据。当绘制的数据图的时间点超过系统所保留的历史数据时间长度时，则系统只能使用“趋势数据”来绘制相应的数据图。这个道理其实很简单，例如，当将某个监控项目配置成最长保留两天的历史数据，而要求系统绘制对应监控项目三天前的数据图时，则系统只能用“趋势数据”来绘制，因为三天前的历史数据已经被删除了。
- 是否使用“趋势数据”来绘制数据图，还与所需要绘制的数据图上数据点的密集程度有关。如果在数据图的水平方向上，每个像素所代表的时间长度超过 225 秒的话，则系统将会自动用“趋势数据”来绘图，而不管所对应的时间点上“历史数据”是否可用。这一点其实也很好理解，当在屏幕水平方向上一定数量的像素长度内，如果所需要绘制的数据点数达到某个值时，系统将无法分辨并绘制它。这其实就是分辨率的问题。

5.4.3 网卡流量数据图配置

与项目样板类似，在一个自动发现规则上，也可以创建数据图样板。而在一个被监控主机上创建数据图与在一个自动发现规则上创建数据图样板，其操作方法和操作流程是类似的。所以，接下来，我们在前面所创建的“发现系统网卡”自动发现规则上，创建一个数据图样板，并介绍如何在 Zabbix 系统中创建数据图。

依次选择菜单项“系统配置”→“模板”，在“配置模板”列表页面上找到前面所创建的模板“test1”，并单击对应行“自动发现”字段上的超链接。接下来，在“配置自动发现规则”列表页面上，找到刚刚配置的“发现系统网卡”这个自动发现规则，并单击对应行“数据图样板”字段上的超链接。最后，在“配置数据图样板”页面上，单击页面右上部的“新建数据图样板”按钮，系统将打开如图 5-18 所示的“配置数据图样板”表单页面。

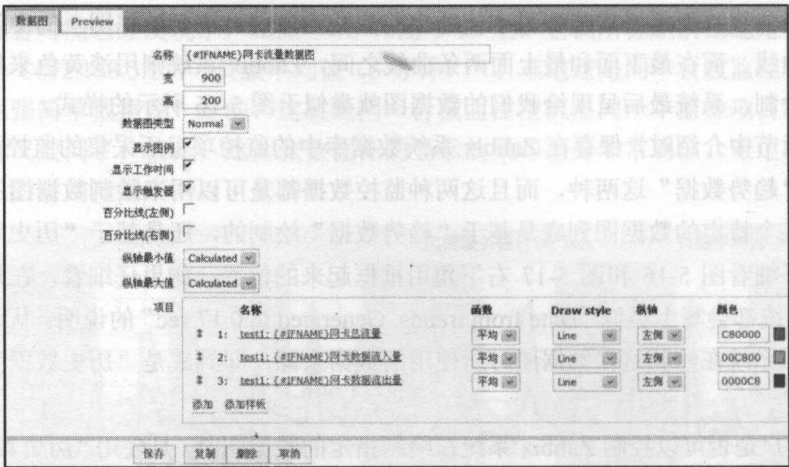


图 5-18 配置数据图样板表单页面截图

从图 5-18 可以看出，在配置数据图样板表单页面上，有两个选项卡，它们分别是“数据图”和“预览（Preview）”选项卡。其中，“预览”选项卡是用来给我们预览数据图效果的，该选项卡上没有任何表单项需要我们填写。而“数据图”选项卡上的表单项大抵可以分成两个区域，即用于配置数据图属性的表单区域和用于配置数据图项目的“项目”区域。其中，用于配置数据图属性的表单区域内各表单项及它们的含义和作用如表 5-2 所示。

表 5-2 配置数据图属性表单项列表

表单项名称	描述
名称（Name）	用于配置数据图名称。数据图名称可以支持宏变量，如图5-18所示，我们所配置数据图的名称即为“{#IFNAME}网卡流量数据图”
宽（Width）	用于指定数据图的宽度，单位是像素。但是这个参数仅对预览功能和饼形图及切开的饼形图有效
高度（Height）	用于指定数据图的高度，单位是像素
数据图类型 （Graph type）	<div>用于指定要创建的数据图的类型。Zabbix系统中支持创建以下类型的数据图：</div> <ul style="list-style-type: none">● 一般数据图（Normal），即一般意义上的曲线图。● 层叠图（Stacked），即靠用不同颜色填充区域的高低来反映数值大小的数据图。但是，需要注意的是，它与柱形图是两种不同类型的数据图。为使读者对层叠图有直观的印象，这里截了一个层叠图： <div></div> <ul style="list-style-type: none">● 饼形图（Pie），这种图比较常见，相信读者都见到过，无须我们在这里作过多的解释● 切开的饼形图（Exploded），这种数据图与上面所述的饼形图是类似的，只是表示不同项目数值大小的各个不同颜色区域是相互分开的
显示图例 （Show legend）	用于指定在所生成的数据图上是否显示图例
显示工作时间 （Show working time）	如果选中这个复选框，则系统在显示数据图时，非工作时间区间内的数据图部分将显示成灰色背景，而工作时间内的数据图部分，将显示成白色背景。但是，这个表单项对于饼形图和切开的饼形图无效


表 5-2 配置数据图属性的各表单项的作用和含义

续表

表单项名称	描述
显示触发器 (show triggers)	如果选中这个复选框，则系统在显示数据图时，会将与对应数据图所引用的监控项目相关的触发器信息显示在数据图的底部
百分比线（左侧）	如果选中该复选框，则系统在显示数据图，且纵坐标轴被设置在数据图的左侧时，会在数据图上指定的位置画一条水平线。而这条水平线所画的位置，则是由该表单项里所输入数值以及数据图所引用的数据共同决定的。例如，当在这个表单项里输入95时，则这条水平线将会被画在，使数据图所引用的所有数据中，有95%在这条水平线之下的位置
百分比线（右侧）	这个复选框的作用与“百分比线（左侧）”复选框的作用是一样的，只是它是针对纵坐标轴在数据图右侧的情况的
纵轴最小值 (Y axis MIN value)	用于指定纵坐标轴所显示的最小值。这个表单项有三个可选项： <ul style="list-style-type: none">● 通过计算获得（Calculated），即通过对数据图所引用的数据进行计算，从而估算出纵轴的最小值● 固定值（Fixed），即，指定一个固定的纵轴最小值● 项目（Item），即以指定的监控项目所采集的最后一个监控数据值，作为纵轴的最小值。 需要说明的是，该表单项里的内容，我们一般选择“通过计算获得”这个选项。如果使用固定值，那么当绘图所使用的实际数据小于这里所指定的最小值时，则这部分数据所对应的数据图将无法绘出
纵轴最大值 (Y axis MAX value)	这个表单项与“纵轴最小值”表单项的作用是类似的，只是这个表单项是用于指定纵轴最大值的。且该表单项也有“通过计算获得（Calculated）”、“固定值（Fixed）”和“项目”三个备选项
3D显示（3D view）	如果选中这个复选框，则数据图将会以3D的样式进行显示。该表单项只对饼形图和切开的饼形图有效

表 5-2 所列出的是配置数据图属性的各表单项的作用和含义。我们在“名称”表单项里输入“{#IFNAME}网卡流量数据图”，并勾选上“显示图例”、“显示工作时间”和“显示触发器”等几个多选框后，就完成了数据图属性部分的配置。而配置数据图属性的其他表单项可以保持默认值。接下来，需要在“项目”表单区域配置数据图所引用的监控项目及它们的属性。表 5-3 列出了“项目”表单区域中各表单项的含义和作用。

表 5-3 “项目”表单区中表单项列表

表单项名称	描述
排序顺序 (Sort order)	用于指定对应监控项目的绘图次序。次序的范围为0~100。排在前面的监控项目，即次序号为0的监控项目将被最先绘图。可以用鼠标选中次序号前面的“  ”图标，并拖动对应的监控目，来改变监控项目的绘图次序
名称（Name）	显示的是监控项目或项目样板的名称
类型（Type）	该表单项只对饼形图或切开的饼形图有效。实际上，当将数据图配置成其他类型的数据图时，这个表单项是不会显示出来的。这个表单项有以下两个备选项：简易型（Simple），当选这个备选项时，系统会以监控项目的数据在总数据中的比率来绘制饼形图。例如，图5-18所示的我们所创建的网卡流量数据图，总共选择了“{#IFNAME}网卡总流量”、“{#IFNAME}网卡数据流入量”和“{#IFNAME}网卡数据流出量”三个项目样板。假如这三个项目样板在某台被监控主机上对应的实际监控项目（不妨称它们为监控项目A、B、C），在某个时刻所采集到的监控数据分别为50kb/s、20kb/s和30kb/s。此时，当A、B、C所对应的数据图

续表

表单项名称	描述
	<p>样板类型都选择“简易型”时，则监控项目A在饼形图上所占面积为50%，而监控项目B和监控项目C在饼形图上所占的面积分别为20%和30%</p> <ul style="list-style-type: none"> ● 总计型（Graph sum），如果数据图所引用的多个监控项目中，有一个监控项目被选择为总计型的，那么该监控项目所采集的数据就是饼形图整个面积所对应的数据。仍然以上述的数据为例，如果将A监控项目设置为“总计型”，那么监控项目B和监控项目C在饼形图上所占的面积就分别为：20/50=40%和30/50=60%。而这个时候，饼形图上将无法显示监控项目A的图形
函数（Function）	<p>用于指定，如果一个监控项目有多个数据可用时，系统使用哪种数据作为绘图的依据。备选项有：</p> <ul style="list-style-type: none"> ● 全部（all），表示使用全部可用的数据 ● 最小值（MIN），表示使用数据集中的最小值 ● 平均值（AVG），表示使用数据集的平均值 ● 最大值（MAX），表示使用数据集中的最大值
绘图样式（Draw style）	备选项有“实线（Line）”、“填充区域（Filled region）”、“粗实线（Blod line）”、“点线（Dot）”和“虚线（Dashed line）”等
纵轴位置（Y axis side）	用于指定系统在绘图时，将纵轴绘制在数据图的左侧还是右侧。备选项有“左侧”和“右侧”
颜色（Colour）	用于指定绘图所使用的颜色。可以输入十六进制的RGB颜色值，也可以单击颜色图标，在系统弹出的调色板上进行选择

单击“项目”表单域的“添加”或“添加样板”上的超链接，可以向数据图中添加所引用的监控项目或项目样板。因为我们在这里所创建的是数据图样板，所以，需要单击“添加样板”上的超链接，并将之前在“发现系统网卡”自动发现规则上所配置的三个项目样板添加进数据图。其他表单项的内容依据你的实际需要进行填写。

完成上述各表单项的填写后，单击页面下部的“保存”按钮。这样就创建好了一个数据图样板。当把 test1 模板关联到某台主机上后过几分钟，依次选择菜单项“状态统计”→“数据图”，就可以查看到如图 5-19 所示的数据图了

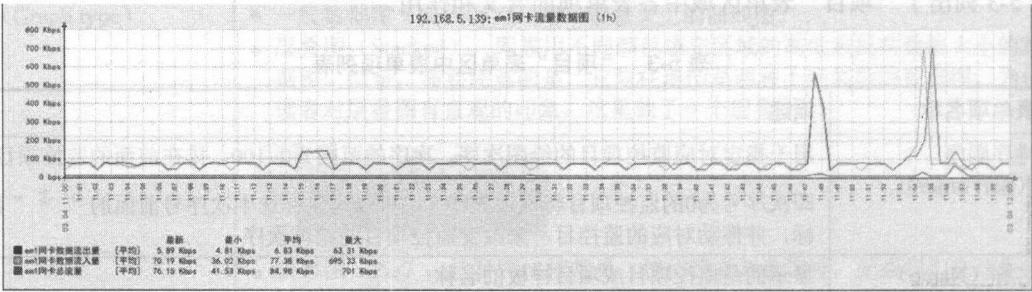


图 5-19 网卡流量数据图

5.5 触发器配置进阶

前面章节对于如何配置触发器做了一些简单介绍，本节将对触发器表达式的编写、触发器的级别以及触发器的依赖等方面的内容做进一步介绍和说明。

5.5.1 触发器计算表达式

在 Zabbix 系统中，可以引用监控项目书写出比较复杂的触发器计算表达式。如果向触发器计算表达式中，代入所引用的监控项目所采集的监控数据后，使得触发器计算表达式计算的结果为逻辑“真”，则触发器就会被触发。简单的触发器计算表达式的格式为：{<被监控主机 IP 或主机名>:<关键字>.<函数>(<参数>)}<操作符><常量>。例如，我们在前面章中介绍创建第一个触发器时，所写的触发器计算表达式{192.168.5.139:system.cpu.load.avg(180)}>1 即为这种格式的计算表达式。

1. 关于函数

在触发器计算表达式中，可以使用大量的 Zabbix 系统所支持的函数，例如 last()、max()、avg()、min()等。要想了解 Zabbix 系统支持哪些函数，可以参阅本书的附录 A。

当在触发器表达式中使用系统函数时，则可以使用监控项目所采集的监控数据、当前时间、常数以及其他数据作为函数的参数。当在函数参数中使用常数时，则这个常数在系统中将被视为时间，单位是秒。例如，前面所编写的触发器表达式{192.168.5.139:system.cpu.load.avg(180)}，即表示计算被监控主机 192.168.5.139 上，“system.cpu.load”这个关键字所对应的监控项目，从当前时间的前三分钟开始到现在，所采集到的所有监控数据的平均值。而如果函数参数数字前带上井号，则表示取数据个数。例如，如果把上面这个触发器表达式修改成{192.168.5.139:system.cpu.load.avg(#180)}，则表示针对被监控主机 192.168.5.139 上，“system.cpu.load”这个关键字所对应的监控项目，所采集的最近的 180 个数据进行求平均值。但是，上述表示方法在 last()函数中稍有不同。当在 last()函数中使用“#数字”形式的参数时，则表示取倒数第“数字”个数据。例如，当把上述的触发器表达式修改成{192.168.5.139:system.cpu.load.last(#2)}时，则表示取被监控主机 192.168.5.139 上，“system.cpu.load”这个关键字所对应的监控项目所采集的倒数第二个监控数据。

函数 avg、count、last、min 和 max 等，还支持第二参数，即时间偏移参数。如果在这些函数中使用时间偏移参数，则表示开始计算的时间点向前推移时间偏移参数所指定的时间长度。例如，avg(1h,1d)表示计算一天前的当前时刻开始，一个小时内所有数据的平均值。

在触发器计算表达式中，可以使用 Zabbix 系统支持的单位符号。例如，“5m”表示 5 分钟，即 300 秒；而 1d 则表示一天，即 86400 秒。Zabbix 系统支持的单位符号列表，请参阅本书的附录 B。

提示：1. 在 Zabbix 系统中，只能将“历史数据”代入到触发器表达式中进行计算，“趋势数据”不能被代入到触发器表达式中进行计算。所以，在书写触发器表达式，特别是当表达式所调用的函数使用了时间偏移参数时，一定要保证在偏移了指定的时间长度后，系统历史数据表中仍有相关项目的监控数据。否则，对应的触发器将会变成不被系统支持的状态。

2. 在触发器表达式中，每个被调用的函数至少需要带一个参数，即使该参数在实际计算中没有使用上，例如 last(0)等。

2. 关于运算符

触发器表达式所支持的运算符及它们的计算优先级如表 5-4 所示。

表 5-4 运算符列表

优先级	运算符	描述
1	/	除
2	*	乘以
3	-	减号
4	+	加号
5	<	小于
6	>	大于
7	#	不等于
8	=	等于
9	&	逻辑与
10		逻辑或

5.5.2 关于触发器依赖

我们知道，在实际监控环境中，经常会遇到这样一种情况，一种服务，或者说一种被监控对象的状态会依赖于另一种服务或被监控对象的状态。例如，Web 服务器上 Web 端口的状态会依赖于该服务器主机的存活状态。通俗一点说，要保证某台 Web 服务器上 80 端口状态是正常的，则首先要保证这台主机上操作系统是正常启动和运行的。如果这台主机本身就已经宕机了，那么它上面所运行的 Web 服务肯定也会宕机，其所对应的端口也肯定不正常。再比如，某台路由器的后端带了许多服务器，那么当跨网段检测这台路由器后端主机的存活状态时，则首先需要保证这台路由器是工作正常的。如果路由器都宕机了，那么当 Zabbix 系统通过它来检查它后面所带的主机的存活状态时，则这些主机的存活状态肯定是不正常的。同样，在这种情况下，当用户通过这台路由器访问它后端主机上的服务时，肯定也是不正常的。

在监控如上述这种具有关联关系的多种服务的状态时，如果能在监控系统中反映出这种关联关系，并且在具有关联关系的多个服务同时出现故障时，只对最基础的服务进行报警，则不但可以避免发送很多不必要的报警信息，同时也可以大大减少由此带来的误判。例如，在一台主机的 Web 端口状态和主机存活状态同时出现故障时，系统只需对主机的存活状态进行报警，而只有在将主机存活状态恢复正常了，但 Web 端口仍然处于故障状态时，系统才会针对 Web 端口的故障进行报警。同样，如果路由器和它后端所带的主机同时出现故障，则只有在路由器的故障恢复以后，它后端所带的主机仍然处于故障状态时，系统才会针对相应的主机进行报警。否则，如果某台路由器后端带了几十台甚至数百台服务器，而一旦路由器或者网络链路因为某种原因而出故障，系统需要发出的报警信息量将是非常可观的。这样，不但对资源是不必要的浪费，而且对于相关人员，一下子接收到数百封甚至上千封邮件或短信，那也将是一件十分恐怖的事情。

针对上述问题，如果在监控系统中针对不同主机上不同服务之间引入某种关联性，或者说依赖性，就可以很好地解决该问题。在 Zabbix 系统中，我们虽然不能直接在主机之间建立某种依赖性。但是，Zabbix 系统为我们提供了一种更灵活的方法，那就是，可以在不同的触发器之间建立起依赖关系。例如，在上面的例子中，如果在反映 Web 端口状态的触发器（下面我们简

称触发器 A) 上, 建立起与反映主机活动状态的触发器 (以下简称触发器 B) 之间的依赖关系。此时, 当触发器 B 进入到“问题”状态时, 也就是主机宕机时, 很显然 Web 服务的端口状态也应该是不正常的, 而且也会被 Zabbix 系统检测到。但是, 因为在触发器 A 上建立起了对触发器 B 的依赖, 所以, 触发器 A 的状态并不会跟着立即转变成“问题”状态, 而是仍然保持“正常”状态。因为触发器 A 并没有从“正常”状态转变为“问题”状态, 所以, 在系统中也就不会产生相应的事件, 从而也就不会触发相关动作, 因此, 也就不会给用户发送有关 Web 端口状态不正常的报警信息。类似的, 当在反映主机存活状态的触发器 B 上建立起对反映路由器状态的触发器 C 的依赖关系后。则, 只要触发器 C 处于“问题”状态, 那么不管主机存活状态是否是“问题”状态, 反映它的触发器的状态也不会从“正常”状态转变为“问题”状态。既然触发器的状态没有改变, 那么在 Zabbix 系统中也就不会产生相应的事件。没有相应事件的产生, 也就更谈不上触发相关动作了。从而, 也就避免了因为路由器或者网络链路的故障, 导致 Zabbix 系统一下子给用户发送几百甚至上千封邮件或短信的情况发生。

从上面的叙述中可以看出, 我们对 Zabbix 系统中触发器之间依赖关系的理解, 应与平常我们对“依赖”这个词的理解稍有不同。按照我们平常对“依赖”这个词的理解, 如果触发器 A 依赖于触发器 B, 则说明只有触发器 B 被触发了, 触发器 A 才有可能被触发。也就是, 只是当触发器 B 的状态转变为“问题”状态时, 触发器 A 的状态才有可能变成“问题”状态, 即触发器 B 状态的改变是触发器 A 状态改变的必要条件。如果要是这样理解 Zabbix 系统中触发器之间的依赖关系, 则正好把 Zabbix 系统中触发器之间的依赖性理解反了。在 Zabbix 系统中, 如果触发器 A 依赖于触发器 B, 那么触发器 B 的状态处于“正常”是触发器 A 可能被触发的必要条件。换句话说, 只有触发器 B 没有被触发, 触发器 A 才有可能从“正常”状态转变为“问题”状态。在理解 Zabbix 系统中触发器之间的依赖性时, 这一点需要特别注意。

另外, 在配置和使用 Zabbix 系统中触发器之间的依赖关系时, 还需要注意以下几点:

- 一台主机上某个触发器可以依赖于本主机上的其他触发器, 也可以依赖于其他主机上的任何触发器。但是, 依赖关系不能循环。例如, 如果主机 A 上的触发器 I 依赖于主机 B 上的触发器 II, 那么主机 B 上的触发器 II 不可以再依赖于主机 A 上的触发器 I。
- 一个模板上的触发器可以依赖于本模板上的其他触发器, 也可以依赖于其他模板上的任何触发器。但是, 如果模板 A 上的某个触发器依赖于模板 B 上的某个触发器, 则当将模板 A 被关联到某台主机上 (或其他模板) 时, 模板 B 也必须关联到这台主机 (或模板) 上。相反, 如果模板 B 被关联某台主机 (或其他模板) 上时, 则该主机 (或模板) 不一定非要关联模板 A。
- 一个模板上的触发器可以依赖于某台主机上的触发器。在这种情况下, 当将这个模板关联到其他主机上时, 那么关联了该模板的所有主机上, 依据该模板而创建的相应触发器, 都会依赖于模板中相应触发器所依赖的触发器。例如, 如果模板 A 上的触发器 I 依赖于路由器上的触发器 II, 那么所有关联了模板 A 的主机上, 依据模板 A 所创建的触发器 I 都依赖于路由器上的触发器 II。
- 一个触发器可以依赖于多个其他触发器。
- 主机上的触发器 (是指直接创建在主机层面上的触发器), 不可以依赖于任何模板上的触发器, 但是, 可以依赖于主机依据模板在特定主机上所创建的触发器。例如, 主机 A 上的触发器 I, 不可以直接依赖于模板 B 上的触发器 II。但是, 如果将模板 B

关联到了主机 C 上，且在主机 C 上依据模板 B 创建了触发器Ⅲ，那么，主机 A 上的触发器 I 就可以依赖于主机 C 上的触发器Ⅲ。

5.5.3 关于触发器级别

在 Zabbix 系统中，触发器的级别从低到高共分六级。对这些级别的名称和它们在页面上显示时所使用的颜色，作为用户的我们可以修改和自行定义的。触发器级别列表如表 5-5 所示。

表 5-5 触发器的级别列表

级别	默认名称	建议中文名称	描述	默认颜色
0	Not classified	未分类信息	未知的错误	灰色 (DBDBDB)
1	Information	一般信息	一般意义上的提示	淡绿色(D6F6FF)
2	Warning	警告信息	普通的警告	淡黄色(FFF6A5)
3	Average	重要问题	比较重要的警告	橘色(FFB689)
4	High	严重问题	严重的问题	淡红色(FFB689)
5	Disaster	灾难问题	灾难性的问题	深红色(FF3838)

- Zabbix 系统中不同的触发器级别，可以应用在以下这些场合：
- 显示触发器列表或状态时，系统使用不同的颜色来区分不同的触发器级别。
 - 当有报警时，系统在 Web 前台会通过声音来提示用户。此时，被触发的触发器级别不同，系统提示用户的声音也不同。
 - 可以应用于用户消息介质上，定义不同的消息介质用于发送不同级别的报警信息。例如，手机短信消息介质只用于发送“严重问题”以上级别的报警消息，而邮件消息介质则可以用于发送“一般信息”级别以上的报警信息。关于配置不同的消息介质发送不同级别报警信息的方法，我们将在后续章节中详细介绍。
 - 触发器的级别，可以作为动作是否被“触发”的条件要素。

修改触发器级别名称和显示颜色的方法很简单，依次选择“高级配置”→“常规”菜单项，在“配置图形界面”页面右上部的下拉菜单里选择“触发器报警级别”菜单项，系统将打开如图 5-20 所示的配置触发器级别的页面。

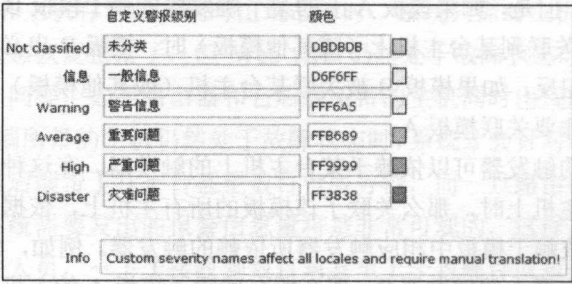


图 5-20 配置触发器级别表单页面截图

5.5.4 配置磁盘分区空间使用率触发器

读者大概还记得，在前面章节中介绍如何配置模板时，在系统中创建了一个叫作 test1 的模板，并且后来所配置的所有自动发现规则都创建在这个模板上。同时，在创建 test1 模板时，在

这个模板上创建了两个模板级的宏变量{ \$LOWDISKUSAGE }和{ \$HIGHDISKUSAGE }。其中，宏变量{ \$LOWDISKUSAGE }指代的值为 60，而宏变量{ \$HIGHDISKUSAGE }指代的值为 80。接下来介绍如何在 test1 模板的“发现系统磁盘分区”自动发现规则上创建触发器样板。

依次选择菜单项“系统配置”→“模板”，在“配置模板”列表页面上找到前面所创建的模板“test1”，并单击对应行“自动发现”字段上的超链接。接下来，在“配置自动发现规则”列表页面上，找到我们在前面所配置的“发现系统磁盘分区”这个自动发现规则，并单击对应行“触发器样板”字段上的超链接。最后，在“配置触发器样板”页面上，单击页面右上部的“新建触发器样板”按钮，系统将打开如图 5-21 所示的“配置触发器样板”表单页面。

如图 5-21 所示，在“名称”表单项里输入：{ #SNMPVALUE } 磁盘空间使用率大于 { \$LOWDISKUSAGE }；在“表达式”表单项里输入：{ test1:hrStorageUsage[{ #SNMPVALUE }].last(0) } > { \$LOWDISKUSAGE }；“警报级别”选择：警告信息，并勾选“启用”复选框，并单击“保存”按钮。这样，就创建好了一个触发器样板。

其中，宏变量{ #SNMPVALUE }是在自动发现规则中定义的。当该模板被关联到某台主机上后，宏变量将会被实际系统所发现的分区名称替换。宏变量{ \$LOWDISKUSAGE }是在前面创建模板时，在模板层面上定义的，它的值是 60。当在关联了模板 test1 的主机上，再定义一个同名宏变量并赋予不同的值时，就可以实现不同的主机，对于同一类监控项目具有不同的报警阈值。关于这一点，在前面介绍宏变量的替换顺序时做过详细介绍。

名称 { #SNMPVALUE } 磁盘空间使用率大于 { \$LOWDISKUSAGE }

表达式 { test1:hrStorageUsage[{ #SNMPVALUE }].last(0) } > { \$LOWDISKUSAGE } 添加

表达式构造器

Multiple PROBLEM events generation

描述

URL

警报级别 未分类 一般信息 警告信息 重要问题 严重问题 灾难问题

启用 ☒

图 5-21 配置触发器样板表单页面截图

上面创建了一个触发器样板。将 test1 模板关联到了某台主机上后，当系统某个分区空间的使用率超过宏变量{ \$LOWDISKUSAGE }所指定的阈值时，该主机上依据 test1 模板上触发器样板所创建的触发器就会被触发。如果有需要，我们还可以使用另外一个宏变量{ \$HIGHDISKUSAGE }，来配置一个拥有更大触发阈值的触发器样板。其配置方法与上面所介绍的配置方法是一样的，只需将宏变量{ \$LOWDISKUSAGE }换成{ \$HIGHDISKUSAGE }，并将警报级别选择成“严重问题”即可。

当配置两个具有不同触发阈值的触发器，并结合不同的消息介质发送不同级别的报警信息时，就可以实现对于同一个监控对象其所出现问题的程度不同，系统用不同的消息介质发送报警信息的功能。

5.6 动作配置进阶

在前面章节中，对 Zabbix 系统中动作的概念和作用以及动作行为升级的概念等都做过一些简单介绍，并且还动手配置了一个触发器类动作。最后，人为地创造了条件，触发了这个触发器类的动作，并让它给我们发送了第一条报警信息。接下来，对动作的触发条件以及 Zabbix 系统所支持的其他两种类型的动作——“自动发现”和“自动注册”作进一步的介绍和说明。

5.6.1 关于动作分类

我们知道，在 Zabbix 系统中，系统事件主要来源于三个方面（除了 Zabbix 2.2 以后版本中增加的内部事件）：触发器状态的改变、被监控设备代理的自动注册、自动发现的主机或服务器状态的改变等。在 Zabbix 系统中，与这三种事件来源相对应的，有三种类型的动作，它们是触发器（Triggers）类动作、自动发现（Discovery）类动作和自动注册（Auto registration）类动作。

其中，触发器类动作是针对系统中触发器状态改变的，也即事件来源是触发器状态的改变。自动发现类动作是针对 Zabbix 系统中网络自动发现功能的，也即事件来源是网络自动发现。需要说明的是，我们这里所说的网络自动发现功能与前面介绍的低级自动发现功能，并不是一回事。低级自动发现功能主要是针对监控项目或者说具体的被监控对象的，其属于监控项目级别的自动发现。它的主要目的和作用是，解决同一类型的监控项目在不同的被监控主机上具有不同个数的问题。而网络自动发现功能，则是针对被监控主机而言的，是属于被监控主机层面的，它的主要作用和功能是，解决自动发现网络中有哪些存活主机的问题。关于网络自动发现功能更详细的说明与介绍，以及它的配置方法与原理，将在后续章节中作进一步的阐述。

通过前面的介绍，我们知道，在触发器类动作中，通过动作系统可以执行向指定的用户或用户组里的用户发送报警信息，或者在指定的远程主机上执行指定的命令等协作行为。而自动发现和自动注册类的动作，它们可支持执行的行为种类要比触发器类动作所支持的可执行行为种类要多很多。具体来说，自动发现类和自动注册类动作，可以支持执行的行为种类如下所列。

1. 自动发现类动作支持的行为种类

- **自动在系统中添加被监控主机。**当系统执行网络自动发现功能发现一台新的主机时，系统可以自动将所发现的主机添加为被监控主机。
- **自动删除被监控主机。**当系统执行网络自动发现功能发现某台被监控主机丢失或状态不可用时，可以自动将这台主机从系统中删除。
- **启用被监控主机。**如果系统在执行网络自动发现功能时，发现某台之前不可用或丢失的主机，现在又变成成为可用状态了，则可以自动将这台主机的状态由“禁用”变成“启用”。
- **禁用被监控主机。**如果系统在执行网络自动发现功能时，发现某台之前可用的主机，现在变成不可用或丢失状态，则可以自动将这台主机的状态由“启用”变成“禁用”。
- **将主机添加到主机组。**当系统执行网络自动发现功能时，发现了一台新的主机或发现某台主机的状态由“不可用”变成了“可用”时，系统可以将对应主机添加到指定的主机组中。
- **将主机从主机组中删除。**如果系统在执行网络自动发现功能时，发现某台之前可用的

主机，现在变成不可用或丢失状态，则可以自动将这台主机从指定的主机组中移出。

- **将指定模板关联到主机。**当系统在执行网络自动发现功能时，发现一台新的主机或发现某台主机的状态由“不可用”变成“可用”时，系统可以自动将指定的模板关联到对应主机上。
- **取消模板关联。**如果系统在执行网络自动发现功能时，发现某台之前可用的主机，现在变成不可用或丢失状态，则系统可以自动取消指定的模板与该主机的关联。

2. 自动注册类动作支持的行为种类

- **增加主机。**当有新的主机通过被监控设备代理自动注册到 Zabbix 服务器端时，系统可以自动地将这台主机添加为被监控主机。
- **禁用主机。**在某些时候，Zabbix 服务器端可能会收到来自被监控设备代理发来的自动注册信息，但是对于这些主机我们可能并不需要立即对它们监控。此时，就可以让系统将此类主机添加为被监控主机，但是将它们的状态设置成禁用状态。因为，系统并不知道这些自动被添加的但是被禁用了的主机什么时候可以被启用，所以对于这类被禁用的主机，需要通过手工在需要的时候启用它们。
- **将主机添加到主机组。**当有新的主机通过被监控设备代理自动注册到 Zabbix 服务器端时，系统可以自动地将这台主机添加到某个或某些主机组中。
- **关联指定的模板到主机。**当有新的主机通过被监控设备代理自动注册到 Zabbix 服务器端时，系统可以自动地将某些指定的模板关联到该台主机上。这个时候，被自动注册的主机，就会变成系统中的一台被监控的主机了。

以上详细地列出了自动发现类动作和自动注册类动作所能支持的行为种类以及功能，灵活和巧妙地使用这些行为，不但能大大地减小我们日常维护和管理监控系统的工作量，而且也可以大大地增强 Zabbix 系统的智能化和自动化，并能大大地减少因为手工操作而带来的人为失误。但是，在配置和使用上述这些行为时，我们还需要注意以下几点：

- 在执行网络自动发现规则时，系统并不是将对应的网络自动发现规则执行完成之后，才去执行相应的动作和行为的。而是，每发现一台新主机，或发现某台之前被发现的主机不可用或丢失时，就会立即执行相应的动作和行为。
- 上述这些自动发现类动作行为和自动注册类动作行为，其所操作的对象只能是被发现（或自动注册）的主机（或者是丢失的主机）。动作行为无法做到因为自动发现了主机 A，所以将主机 B 添加到系统中或者从系统中删除。即所添加或删除的对象，只能是被发现的主机 A，而不可以是其他第三方主机。
- 当通过网络自动发现功能自动向系统中添加主机时，所被添加主机的主机名是由系统调用 `gethostbyname` 函数，根据被发现主机的 IP 地址反向解析获取的。如果系统调用函数 `gethostbyname`，根据被发现主机的 IP 地址能反向解析出所发现主机的主机名，则这个主机名就被作为被发现主机的主机名。否则，系统将使用被发现主机的 IP 地址作为它的主机名。而如果被发现主机的 IP 地址是 IPv6 地址，那么，系统将会自动使用下划线（`_`）来替换对应 IP 地址中的冒号（`:`），并将替换后的内容作为被发现主机的主机名。这么做的原因是，在 Zabbix 系统中，主机名中是不可以带有冒号的。
- 如果网络自动发现规则是由 Zabbix 代理服务器来执行的，且发现了新的主机，则对主机的主机名的解析工作，仍然由 Zabbix 服务器来执行。因为 Zabbix 服务器和 Zabbix 代理服务器所配置的 DNS 服务器可能不同（或其他原因），可能会出现通过网络自

自动发现功能所发现的主机的 IP 地址，在 Zabbix 代理服务器上能解析出主机名，而在 Zabbix 服务器上却解析不出主机名。这个时候，被发现的主机在 Zabbix 系统中的主机名只能是 IP 地址形式。

□ 如果通过网络自动发现功能所发现的某台主机，在 Zabbix 系统中存在相同主机名的主机（但是 IP 地址不同）。此时，Zabbix 系统在自动向系统中添加被发现的主机时，会自动在主机名后面添加上“_N”形式的内容，以免与系统中已存在的主机重名。其中，N 为从 2 开始的自增长整数。

5.6.2 关于动作触发条件

在前面章节中，我们详细介绍了“配置动作”页面的“动作”和“行为”选项卡上，各表单项的作用和含义。在本节，我们将来介绍一下“配置动作”页面的另外一个选项卡——“触发条件（Conditions）”上各表单项的作用和含义。“配置动作”页面的“触发条件”选项卡上的表单项如图 5-22 所示。

图 5-22 动作“触发条件”选项卡表单页面截图

我们知道，虽然事件可以驱动动作触发。但是，当在 Zabbix 系统中有事件产生时，是不是会有相应的动作被触发，还要看系统中有没有动作的触发条件能匹配得上该事件，只有有动作的触发条件与事件的属性匹配上了，才有可能有动作被触发。所以，动作的触发条件的配置也很重要，如果系统中动作的触发条件配置得不合理，那么就有可能出现系统中虽然已经产生了非常重要的事件，而整个系统中却没有相应的动作去响应它的情况发生。图 5-22 我们截的是触发器类动作触发条件的配置界面，其他两种类型动作的触发条件的配置方法与它基本相同，只是自动注册类动作触发条件不需要配置“计算方式”。因此，如果掌握了一种类型动作触发条件的配置，其他两种类型动作触发条件的配置也就可以轻松掌握了。

从图 5-22 可以看出，整个“触发条件”选项卡页面分成三块表单区域。第一块是选择“计算方式”的表单区域，它实际上是一个下拉菜单。通过它，可以指定，将要新建的触发条件参与逻辑计算的计算方法。这个下拉菜单共有三个备选项，它们是：

- 与（AND）。逻辑“与”计算，即新增加的触发条件与已有的触发条件作逻辑“与”计算。
- 或（OR）。逻辑“或”计算，即新增加的触发条件与已有的触发条件作逻辑“或”计算。
- 与/或（AND/OR）。逻辑“与”/“或”计算。我们知道，在逻辑计算中，要么是“与”计算，要么是“或”计算。那么这里的“与/或”是什么意思呢？当选择这个备选项时，

则实际的计算方法可能是以下两种方法中的一种。当所增加的触发条件与已有的触发条件属于同一类型的触发条件时，则系统会将新增加的触发条件与原有的触发条件作“或”计算，否则将做“与”计算，如图 5-22 所示的触发条件 D 和 E，即因为是同一类型的触发条件，所以它们之间是做“或”计算。而触发条件 A、B 和 C，因为互为不同类触发条件，所以，它们之间是做逻辑“与”计算。

“触发条件”选项卡页面上的第二块表单区域显示的是，当前动作已经配置的触发条件。选中已存在触发条件名称前面的复选框，然后单击该表单区域内的“删除选中项”超链接，则可以将已经选中的触发条件从当前动作中删除。当从当前动作中删除了某些触发条件后，系统将会自动调整余下的触发条件之间的逻辑计算方法。所以，当从当前动作中删除某些触发条件时，需要确认所留下的触发条件的计算结果是否是我们所期望的。

第三块表单区域是用来添加新的触发条件的。不同类型的动作，可以使用该表单区域内的下拉菜单添加不同种类的触发条件。对于触发器类的动作，它可以被添加的触发条件如表 5-6 所示。

表 5-6 触发器类动作所支持的触发条件列表

条件类型	所支持的运算符	描述
监控分类 (Application)	= Like not like	该触发条件支持等于 (=)、like或not like等运算。当使用等号运算符时，则只有监控分类属于该条件所指定的分类的监控项目，它们所对应的触发器的触发而产生的事件，才会与该条件匹配 当使用like运算符时，则只有那些监控分类中，包含该条件所指定的字符串内容的监控项目，它们所对应的触发器所产生的事件，才能匹配该触发条件 相反，当使用not like运算符时，则只有那些监控分类中，不包含该条件所指定的字符串内容的监控项目，它们所对应的触发器所产生的事件，才能匹配该触发条件
主机组 (Host group)	= ◇	当选择“=”运算符时，则该触发条件只会匹配那些与属于指定主机组中的主机相关的触发器的触发所产生的事件 当选择“◇”运算符时，则该触发条件只会匹配那些，与不属于指定主机组中的主机相关的触发器的触发所产生的事件
主机模板 (Host template)	= ◇	当选择“=”运算符时，则该触发条件只会匹配那些，从指定模板中继承过来的触发器所产生的事件 当选择“◇”运算符时，则该触发条件只会匹配那些，不是从指定模板中继承过来的触发器的触发所产生的事件
主机 (Host)	= ◇	当选择“=”运算符时，则该触发条件只会匹配那些，由于指定主机上的触发器的触发所产生的事件 当选择“◇”运算符时，则该触发条件只会匹配那些，不是指定主机上的触发器的触发所产生的事件
触发器 (Trigger)	= ◇	当选择“=”运算符时，则该触发条件只会匹配那些，由于指定触发器的触发所产生的事件 当选择“◇”运算符时，则该触发条件只会匹配那些，不是由该条件所指定的触发器的触发所产生的事件

续表

条件类型	所支持的运算符	描述
触发器名称 (Trigger name)	like not like	当选择“like”运算符时，则该触发条件只会匹配那些，名称中包括该条件所指定字符串的触发器的触发所产生的事件 当选择“not like”运算符时，则该触发条件只会匹配那些，名称中不包括该条件所指定字符串的触发器的触发所产生的事件 需要注意的是，对触发器名称进行匹配时，是区分大小写的。而且，如果触发器名称中包含宏变量，则系统将会使用该触发条件中所指定的字符串与宏变量被替换后的触发器名称进行匹配
触发器级别 (Trigger severity)	= <> >= <=	当选择“=”运算符，则只有那些级别与该触发条件所指定的级别相同的触发器的触发所产生的事件，才会与该触发条件成功匹配 与此类似，当使用“<>”、“>=”或“<=”运算符时，则只有那些级别不等于、大于等于或小于等于该触发条件所指定级别的触发器的触发所产生的事件，才会与该触发条件匹配成功
触发器的值 (Trigger value)	=	当触发器的值等于触发条件所指定的值时，它所产生的事件才会匹配该触发条件 需要注意的是，当一个触发器的状态从“正常”转变为“问题”时，它的值为PROBLEM；而如果一个触发器的状态从“问题”转变为“正常”，则它的值为OK
时间区间 (Time period)	in not in	当选择“in”运算符时，则只有产生时间在触发条件所指定的时间区间内的事件，才会与该触发条件匹配成功 相应的，当选择“not in”运算符时，则只有产生时间不在触发条件所指定的时间区内间的事件，才会与该触发条件匹配成功
维护状态 (Maintenance status)	in not in	当选择“in”运算符时，则只有处在维护状态下的主机上的触发器的触发，所产生的事件才会成功匹配该触发条件 当选择“not in”运算符时，则只有不是处在维护状态下的主机上的触发器的触发，所产生的事件才会成功匹配该触发条件 需要注意的是，如果产生事件的触发器引用了多个主机上的监控项目，则只要这些主机中有一台主机处在维护状态下，那么该触发器的触发所产生的事件即可成功匹配使用“in”运算符所运算的触发条件。相应的，这些主机中只要有一台主机不是处在维护状态下，那么该触发器的触发所产生的事件即可成功匹配使用“not in”运算符所运算的触发条件

表 5-6 列出了触发器类动作可以支持的触发条件以及它们的运算符，并对各种运算符的含义和作用作了说明。在我们创建一个新的动作时，系统会自动为我们在这个动作上添加上‘维护模式状态 not in “维护”’和‘触发器状态=“PROBLEM”’两个触发条件。这是因为，在大多数情况下，当一台主机处于维护状态时，我们并不希望有动作被触发。类似的，在大多数情况

下，我们只关注一个服务或主机从正常状态转变为不正常状态所对应的事件。而对于服务或主机从不正常状态转变为正常状态所对应的事件，只有极少数我们会关注。所以，在我们创建新的动作时，系统会自动添加上‘触发器状态="PROBLEM"'这个触发条件。接下来，表 5-7 列出了自动发现类动作所支持的触发条件及它们所支持的运算符信息。

表 5-7 自动发现类动作支持的触发条件列表

条件类型	支持的运算符	描述
主机IP地址（Host IP）	= ◇	当选择“=”运算符时，则当被发现主机的IP地址在触发条件所指定的IP地址范围内时，该触发条件被成功匹配 当选择“◇”运算符时，则当被发现主机的IP地址不在触发条件所指定的IP地址范围内时，该触发条件被成功匹配
服务类型（Service type）	= ◇	当选择“=”运算符时，则当被发现的服务的类型属于触发条件所指定的服务类型时，该触发条件被成功匹配 当选择“◇”运算符时，则当被发现的服务的类型不属于触发条件所指定的服务类型时，该触发条件被成功匹配
服务端口（Service port）	= ◇	当选择“=”运算符时，则当被发现的服务的端口号属于触发条件所指定的端口号范围内时，该触发条件被成功匹配 当选择“◇”运算符时，则当被发现的服务的端口号不在触发条件所指定的端口号范围内时，该触发条件被成功匹配
发现规则（Discovery rule）	= ◇	当选择“=”运算符时，则当被发现的主机或服务是用触发条件中所指定的发现规则发现的时，该触发条件被成功匹配 当选择“◇”运算符时，则当被发现的主机或服务不是用触发条件中所指定的发现规则发现的时，该触发条件被成功匹配
发现方法（Discovery check）	= ◇	当选择“=”运算符时，则当被发现的主机或服务是用触发条件中所指定的发现方法发现的时，该触发条件被成功匹配 当选择“◇”运算符时，则当被发现的主机或服务不是用触发条件中所指定的发现方法发现的时，该触发条件被成功匹配
发现的对象类型（Discovery object）	=	当被发现的主机或服务的类型是触发条件中所指定的对象类型时，该触发条件被成功匹配
发现状态（Discovery status）	=	Up——当该触发条件被设置为Up时，则用于匹配主机或服务可用事件 Down——当该触发条件被设置为Down时，则用于匹配主机或服务宕机事件 Discovered——当该触发条件被设置为Discovered时，则用于匹配主机或服务被发现事件 Lost——当该触发条件被设置为Lost时，则用于匹配主机或服务丢失事件
上线/下线时间（Uptime/Downtime）	>= <=	上线时间是应用于主机或服务上线事件的，而下线时间则是应用于主机或服务下线事件的 当选择“>=”运算符时，则如果主机或服务的上线或下线时间大于等于触发条件中所指定的秒数时，则该触发条件被成功匹配

续表

条件类型	支持的运算符	描述
		当选择“<=”运算符时，则如果主机或服务的上线或下线时间小于等于触发条件中所指定的秒数时，则该触发条件被成功匹配
接收到的数据 (Received value)	= <> >= <= Like not like	当系统执行自动发现规则时，Zabbix系统会从被发现主机那里接收到相应的数据。因此，“接收到的数据”类型的触发条件，是用于匹配所接收到的数据的 =——当系统所接收到的数据与触发条件中所指定的数据相等时，则该触发条件被成功匹配 <>——当系统所接收到的数据不等于触发条件中所指定的数据时，则该触发条件被成功匹配 >=——当系统所接收到的数据大于或等于触发条件中所指定的数据时，则该触发条件被成功匹配 <=——当系统所接收到的数据小于或等于触发条件中所指定的数据时，则该触发条件被成功匹配 like——当系统所接收到的数据中包含触发条件中所指定的字符串时，则该触发条件被成功匹配 not like——当系统所接收到的数据中不包含触发条件中所指定的字符串时，则该触发条件被成功匹配
服务器代理节点 (Proxy)	= <>	=——当系统所发现的主机或服务是通过触发条件中所指定的Zabbix服务器代理节点发现的时，则该触发条件被成功匹配。 <>——当系统所发现的主机或服务不是通过触发条件中所指定的Zabbix服务器代理节点发现的时，则触发条件被成功匹配。

表 5-7 中，列出了自动发现类动作所支持的触发条件种类以及它们所对应的运算符，并对各种运算符的含义和作用作了说明。与触发器类动作和自动发现类动作相比，自动注册类动作所支持的触发条件类型要少得多，它所支持的触发条件类型及它们所支持的运算符情况如表 5-8 所示。

表 5-8 自动注册类动作所支持的触发条件列表

条件类型	支持的运算符	描述
主机名 (Host name)	= <>	在前面介绍过，当被监控设备代理组件以主动模式运行时，则需要在它的配置文件中配置被监控主机的主机名。在被监控设备代理向Zabbix服务器执行自动注册时，它将使用在其配置文件中所指定的主机名，向Zabbix服务器进行注册 =——自动注册主机的主机名与触发条件中所指定的字符串相等时，则该触发条件被成功匹配 <>——自动注册主机的主机名与触发条件中所指定的字符串不相等时，则该触发条件被成功匹配
服务器代理节点 (Proxy)	= <>	=——当被监控设备代理自动注册是向触发条件中所指定的Zabbix服务器代理节点发送请求的，则该触发条件被成功匹配 <>——当被监控设备代理自动注册不是向触发条件中所指定的Zabbix服务器代理节点发送请求的，则该触发条件被成功匹配

在以上各表详细列出了 Zabbix 系统中各类动作所支持的触发条件类型，以及它们所支持的运算符。在配置和使用这些触发条件时，还需要注意以下几点：

1. 自从 Zabbix 2.0.6 版本开始，如果一个动作的触发条件和行为中所引用的某些监控元素（如主机、模板、触发器等）在系统中被删除了，则该动作中引用了这些元素的触发条件和行为也会被自动删除，而且该动作会被自动设置成“禁用”状态。这样做的目的是，为了避免因为用户删除了某些被引用的元素，而导致执行相关动作产生意外结果。当然，被系统自动禁用的动作，用户是可以通过手工来启用它的。具体来说，当用户在系统中删除以下元素时，系统会自动禁用相关动作，且自动删除对应动作中相关触发条件和行为。

- 删除一个主机组，而该主机组被动作中的触发条件所引用，或在动作行为中被指定为远程命令的执行目标时。
- 删除一台主机，而该主机被动作中的触发条件所引用，或在动作行为中被指定为远程命令的执行目标时。
- 删除一个主机模板，而该模板被动作中的触发条件所引用，或在动作行为中被指定为关联或取消关联的对象时。例如，在某个自动发现类动作的行为中，指定了当系统自动发现一台主机时，将模板 A 自动关联到被发现的主机上。此时，如果将模板 A 从系统中删除，则该动作中的相应行为也会被自动删除，且该动作会被自动设置成“禁用”状态。
- 删除一个触发器，而该触发器被动作中的触发条件所引用时。
- 删除一个自动发现规则，而该自动发现规则和自动发现方法被动作中的触发条件所引用时。
- 删除一个 Zabbix 服务器代理节点，而该代理节点被动作中的触发条件所引用时。

2. 在 Zabbix 2.0.6 之前的版本中，如果用户执行上述操作，则系统不会自动删除相应的触发条件。

3. 如果远程命令的执行目标包含多台主机，那么，当在系统中删除其中一台主机时，系统不会自动删除引用了这台被删除主机的动作行为，而只是将被删除主机从远程命令执行目标列表中删除，除非被删除主机是远程命令执行目标的最后一台主机。同样，当删除模板时，“关联模板”和“取消模板关联”行为也遵守这一规则。

4. 当作为接收报警信息目标的用户或用户组被删除时，对应的动作不会被禁用。

5.6.3 配置清理磁盘空间动作

前面已经配置了一个触发器，用来监控磁盘分区空间使用率。下面，将针对这个触发器创建一个动作。该动作的功能是，在给用户发送报警信息的同时，执行远程命令，以清空目标主机上/opt/nginx/logs/access.log 文件的内容，从而释放磁盘分区上被占用的磁盘空间。

依次选择菜单项“系统配置”→“动作”，单击“配置动作”页面右上部的“新建动作”按钮。系统将打开如图 5-23 所示的“配置动作”表单页面。

配置动作的操作方法和操作过程比较简单，且我们已经在前面章节中详细介绍过如何配置动作。故在此就不再复述配置“清理磁盘空间”动作的每一个详细步骤，仅列出该动作所配置的触发条件和行为，请读者自行完成该动作的配置。

1. 触发条件

维护模式状态 not in “维护” AND 触发器状态=“PROBLEM” AND 触发器名称 包含 “/ 磁盘空间使用率大于 80”。

动作

触发条件

行为

动作行为

步骤 详细

周期(秒) 延迟 动作

1 - 0 发送消息到用户: Admin 默认 立即 编辑

1 Run remote commands on current host 默认 立即 编辑

移除选中项

行为详细

间隔

起始 1

截止 1 (0 - infinitely)

升级周期 0 (最小60秒, 0 - 使用默认值)

行为类型

远程命令

目标列表

目标 动作

Current host 移除

新建

类型

SSH

Authentication method

密码

用户名

root

密码

password

端口

22

命令

>/opt/nginx/logs/access.log

触发条件

没有定义事件

新建

更新 取消

图 5-23 动作配置表单页面截图

2. 行为

在这个动作上，配置两个行为。一个用于发送报警信息，另一个用于在远程目标主机上执行远程命令。

用于发送报警信息的行为配置内容为：

行为类型。发送信息。

发送消息到用户。admin。

默认消息。勾选该复选框。

其他表单项内容，读者可以根据自己的需要，自行填写。

用于执行远程命令的行为的配置内容为：

起始。1

截止。1

行为类型。远程命令。

目标列表。Current host。

类型。SSH。

Authentication method。密码。

命令。>/opt/nginx/logs/access.log。

其他表单项内容，读者可以根据自己的需要，自行填写。

5.7 网络自动发现配置

在前面章节中多次提到过 Zabbix 系统所提供的网络自动发现功能。并且，在前面章节中还详细介绍了网络自动发现事件以及自动发现类动作的配置方法。本节将详细介绍 Zabbix 系统所提供的网络自动发现功能的作用和含义以及它的配置方法。

5.7.1 网络自动发现功能

Zabbix 系统提供的网络自动发现功能，具有非常高效和灵活等特点。通过它，不但可以让 Zabbix 系统自动地查找和发现网络中存活的主机，而且，结合自动发现类动作，还可以很轻松地实现，周期性地查找和发现网络中存活的但是没有被添加到监控系统中的主机，并且自动地将所发现的主机添加到监控系统中，以实现对其进行监控。所以，如果能够用好 Zabbix 中的自动发现功能，不但可以大大提升监控系统的自动化和智能化程度，而且也可以大大地减轻我们日常管理监控系统的工作量，提高工作效率。具体来说，使用 Zabbix 系统中的网络自动发现功能具有以下优势：

- **可以快速部署 Zabbix 系统。**我们知道，在部署和配置监控系统的过程中，向监控系统中添加和配置被监控主机信息、配置监控项目 and 数据图、设置触发器规则等操作，不但工作量巨大，操作起来比较枯燥，而且也是最容易出错的一项工作。但是，如果使用 Zabbix 系统中的网络自动发现功能来完成这些工作，那么，就只需配置好网络自动发现规则和相应的动作规则等，系统就会自动地帮我们完成这些工作。因此，通过网络自动发现功能，可以实现 Zabbix 系统的快速部署。
- **简化监控系统的日常管理操作。**这一点很好理解，当开启 Zabbix 系统的网络自动发现功能，并且配置好网络自动发现规则和相应的动作后，如果监控网络中存在主机数量变动时，比如说监控网络环境中新增或下架了某些主机，则我们几乎不需要对 Zabbix 系统作任何的人工干预。Zabbix 系统中的网络自动发现功能，能自动地发现监控网络中主机的变化，并且能够根据监控网络中主机的这些变化，及时地修改和调整相关配置。因此，使用网络自动发现功能，能够大大减轻我们日常管理和维护监控系统的工作量。
- **能够快速适应监控网络环境的变化。**当需要对监控网络环境做比较大的调整时，比如说机器从一个 IDC 机房搬迁到另一个 IDC 机房，或者因为某种需要，需要对监控网络的结构作出大调整等，都不需要对 Zabbix 系统中的配置做大的调整和修改，而只需修改网络自动发现规则中的相关配置，系统就会自动发现新的监控网络环境中的主机情况，并将它们自动添加到监控系统中。

当 Zabbix 系统执行一个网络自动发现规则时，它将依赖于以下信息来发现监控网络中可能存活的主机：

- **IP 地址范围。**即 Zabbix 系统尝试在哪个 IP 地址范围内，发现可能存活的主机。Zabbix 系统无法自己判断，应该尝试在哪个 IP 地址范围内发现可能存活的主机。因此，需要在自动发现规则中，指定 Zabbix 系统尝试在哪个 IP 地址范围内发现可能存活的主机。
- **外部服务。**可以使用外部服务作为自动发现功能的发现方法，例如 FTP、SSH、Web、POP3、IMAP、TCP 等服务。

□ **被监控设备代理组件信息。**当上面第 2 点所述的外部服务在主机上都不可用时, Zabbix 系统也可以通过检测指定网络中, 主机所提供的被监控设备代理服务状态, 来判断一台主机是否存活。

□ **SNMP 服务信息。**Zabbix 系统可以通过检测指定网络中, 主机所提供的 SNMP 服务状态, 来判断一台主机是否存活。

Zabbix 系统在执行网络自动发现规则并自动在系统中添加所发现的主机时, 会自动在所添加的主机上添加上相应的监控数据采集接口。当系统自动在被添加的主机上添加监控数据采集接口时, 遵循以下规则:

□ 当系统检测到被添加主机上有可用服务时, 将会自动在被添加的主机上添加上与该服务相对应的监控数据采集接口。例如, 如果系统检测到被添加的主机上 SNMP 服务可用, 则其就会自动在该主机上添加上 SNMP 接口。同样, 如果系统检测到被添加主机上被监控设备代理服务可用, 则系统会自动在该主机上, 添加上被监控设备代理数据采集接口。

□ 如果被添加的主机上, 同时有多个服务响应了 Zabbix 服务器的检测, 则系统会将与这些服务相对应的数据采集接口, 都自动添加到该主机上。例如, 如果一台主机同时响应了 Zabbix 服务器所发出的 SNMP 协议请求和被监控设备代理服务请求, 则系统会自动将 SNMP 和 Agent 这两个数据采集接口都添加到这台主机上。

□ 当将通过 SNMP 协议或 Zabbix agent 方式所检测到的数据作为判断主机唯一性标准时, 系统将会将用于发现这台主机的数据采集接口设置成该服务的默认数据采集接口, 其他数据采集接口将会作为该服务的附加数据采集接口, 被添加到对应主机上。

□ 如果某台被添加的主机, 在自动发现它的时候, 只有被监控设备代理服务对 Zabbix 服务器做出了回应, 则系统在自动添加这台主机时, 只会在该主机上添加上被监控设备代理数据采集接口。但是, 此后系统一旦发现该主机上的 SNMP 服务也可用, 则会自动为这台主机添加上相应的 SNMP 数据采集接口。

□ 假如系统最初发现的三台主机, 主机 A、主机 B 和主机 C, 是以 IP 地址作为被发现主机唯一性标准的。但后来发现规则被修改, 改用以通过 SNMP 协议或 Zabbix Agent 方式所采集的数据作为发现这三台主机的唯一性标准, 且此时通过这两种方法所采集的数据是重复的, 则系统会将最先被发现的主机 A 保留, 而另外两台主机 B 和 C, 其所对应的数据采集接口, 将作为附加接口被添加到主机 A 上。相应的主机 B 和主机 C, 将不再作为独立的被监控主机在系统中存在, 即在“系统配置”→“主机”页面上将不再查看到这两台主机。但是, 从“状态统计”→“自动发现”页面上, 仍然可以查看到主机 B 和主机 C 被发现过的记录, 只是它们的记录将以黑体缩进的方式显示在“自动发现设备”栏里。

5.7.2 配置网络自动发现规则

要在系统中创建新的网络自动发现规则, 可依次选择“系统配置”→“自动发现”菜单项。在“配置自动发现规则”页面上, 单击“新建自动发现规则”按钮, 系统将打开如图 5-24 所示的配置自动发现规则表单页面。

名称

ICMP Ping 192.168.5.x

通过代理节点

不通过代理节点

IP地址段

192.168.5.2-253

延迟(秒)

600

检测方式

ICMP ping

SNMPv2 agent "1.3.6.1.2.1.1.0"

新建

检测类型

Zabbix agent

Port range

10050

Key

system.uname

添加

取消

设备唯一性标准

☒ IP地址

☐ SNMPv2 agent "1.3.6.1.2.1.1.0"

启用

☒

图 5-24 配置自动发现规则表单页面截图

图 5-24 中所示的各表单项的含义如表 5-9 所示。

表 5-9 配置自动发现规则表单项列表

表单项	描述
名称 (Name)	用于配置自动发现规则在系统中的唯一名称
通过代理节点 (Discovery by proxy)	指定该自动发现规则是否由某个Zabbix服务器代理节点来执行。可选项有“不通过代理节点”及“系统中所配置的所有Zabbix服务器代理节点列表”。当选择“不通过代理节点”选项时，则该自动发现规则将由Zabbix服务器来执行，否则则由所指定的Zabbix服务器代理节点来执行
IP地址段 (IP range)	用于指定该规则用于发现哪些IP地址范围内可能存活的主机。这个表单项的内容可以按照下列之一的格式进行输入： <ul style="list-style-type: none">● 单个地址形式，例如192.168.5.139● 地址范围形式，例如 192.168.5.1-255，表示系统将查找192.168.5.1到192.168.5.255这个IP地址范围内可能存活的主机● 掩码形式，例如，192.168.5.0/24，表示发现192.168.5.0这个网段内可能存活的主机。但是，需要注意的是，在使用掩码时，Zabbix系统只支持16到32位的掩码● IP地址列表形式，例如，192.168.5.10-100, 192.168.1.1-255, 192.168.2.0/24等需要注意的是，每个被扫描的IP地址只能在所指定的范围里被包含一次。即使不在同一个发现规则里，同一个IP地址也不能被重复包含在不同的地址范围内。如果有IP地址在同一个或不同的发现规则所指定的IP地址范围中被重复包含了，则系统可能会产生意外的错误。例如，系统数据库会被死锁或在系统中产生重复的主机记录等，类似的情况也会发生在两个或多个主机具有多个相同的DNS主机名时
延时 (秒)	用于指定发现规则被执行的时间间隔，单位是秒

续表

表单项	描述
检测方式 (Checks)	指定用何种方法来判断一台主机是否存活。同一个发现规则则可以支持多种检测方式。在Zabbix系统中,支持使用以下服务的状态来检测主机的存活状态:SSH、LDAP、SMTP、FTP、HTTP、POP、NNTP、IMAP、TCP、Zabbix agent、SNMP v1 Agent、SNMP v2 Agent、SNMP v3 Agent和ICMP ping等。除了通过SNMP协议和Zabbix agent检测方式以外,其他的检测方式系统都是通过调用net.tcp.service[]函数来判断相应服务的状态,从而检测对应的主机是否存活的。当使用Zabbix agent检测方式时,系统实际上是通过采集指定监控项目的监控数据,来判断主机的存活状态的。至于采集哪个监控项目的监控数据,则可以由用户自己来指定,附录C中所列的监控项目都是可以作为检测主机存活状态使用的。而当使用SNMP协议方式时,则需要我们指定一个OID值,系统通过查询指定的OID值对象的数据来判断主机存活状态
设备唯一性标准 (Device uniqueness criteria)	指定使用哪种数据作为判断是否是同一台主机的标准 <ul style="list-style-type: none">● IP地址:当使用主机的IP地址作为判断是否是同一台主机的标准时,如果系统发现的主机的IP地址在系统中存在,则系统将其视为已经被发现过的主机,而不会再将该主机作为新的主机添加到系统中● SNMP协议或Zabbix agent方式所检测到的数据:执行发现规则时,系统通过SNMP协议或Zabbix agent方式所采集到的数据作为判断标准
状态 (Status)	禁用或启用

表 5-9 中详细列出了配置自动发现规则表单页面上各表单项的作用和含义。依照表 5-9 的说明,我们在这里创建一个名称为“ICMP Ping 192.168.5.x”的自动发现规则。各表单项的配置参数如下:

名称。ICMP Ping 192.168.5.x。

通过代理节点。不通过代理节点。

IP 地址段。192.168.5.2-253

延时(秒)。600

检测方式为

ICMP ping;

SNMPv2 agent, 其中 OID 为 .1.3.6.1.2.1.1.1.0;

Zabbix agent, 其中 Key 为 system.uname。

设备唯一标准。选择“IP 地址”项。

各表单项内容配置完成后如图 5-24 所示,单击“保存”按钮,保存我们对自动发现规则所作的配置。这样,就创建好了一个自动发现规则。

5.7.3 配置自动发现动作

前面配置了一个自动发现规则,但是,仅有自动发现规则,系统仍然不能自动将所发现的主机添加到系统中并对这些主机实施监控。而要达到这个目的,还需要创建对应的自动发现类动作,通过自动发现类动作来完成向系统中添加被发现主机的功能。

- 使用前面配置的自动发现规则,结合将要配置的自动发现类动作,可以完成以下功能:
- 系统每 10 分钟执行一次发现规则。该时间间隔是通过自动发现规则中延时配置项定义的。

- ❑ 在系统执行自动发现规则过程中，如果发现某台主机的在上线时间超过一个小时，且在系统中没有该主机的配置信息，则系统会自动将该主机添加到系统中。
- ❑ 如果系统发现某台主机的下线时间超过 24 小时，则系统将会自动将该主机从系统中删除。
- ❑ 当系统发现主机上安装的是 Linux 操作系统时，系统自动将该主机添加到“Linux servers”主机组。
- ❑ 当系统发现主机上安装的是 Windows 操作系统时，系统自动将该主机添加到“Windows servers”主机组。
- ❑ 当系统发现主机是 Linux 系统主机时，系统自动将“Template OS Linux”主机模板关联到该主机上。
- ❑ 当系统发现主机是 Windows 系统主机时，系统自动将“Template OS Windows”主机模板关联到该主机上。

因为，在即将要创建的动作中需要用到“Linux servers”和“Windows servers”主机组以及“Template OS Linux”和“Template OS Windows”模板。所以，在创建动作之前，需要按照前面章节中所介绍的方法，自行创建好将要使用的主机组和模板。为了实现 Linux 主机和 Windows 主机分别被添加到不同的主机组中，且关联不同的模板，这里需要创建两个动作。我们不妨将这两个动作分别称作“自动添加 Linux 主机”和“自动添加 Windows 主机”。而要实现，当有主机下线超过 24 小时，系统就自动删除它的功能，则还需要创建一个删除主机的动作，不妨叫它为“自动删除主机”。

创建动作的具体过程和方法，前面章节中已做过详细介绍，在此不再重述这一过程，这里仅贴出每个动作配置表单页面上的“触发条件”和“行为”两个选项卡中所配置内容的截图，请读者根据截图中所示的内容，自行完成相应动作的配置。

1. “自动添加 Linux 主机”动作配置内容

“自动添加 Linux 主机”动作“触发条件”选项卡上的配置如图 5-25 所示。

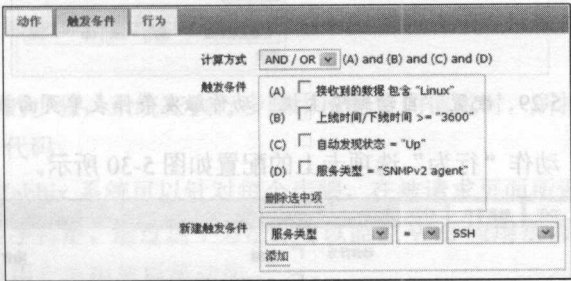


图 5-25 配置“自动添加 Linux 主机”动作触发条件表单页面截图

“自动添加 Linux 主机”动作“行为”选项卡上的配置如图 5-26 所示。

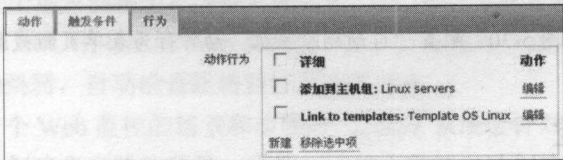


图 5-26 配置“自动添加 Linux 主机”动作行为表单页面截图

2. “自动添加 Windows 主机”动作配置内容

且，“自动添加 Windows 主机”动作“触发条件”选项卡上的配置如图 5-27 所示。

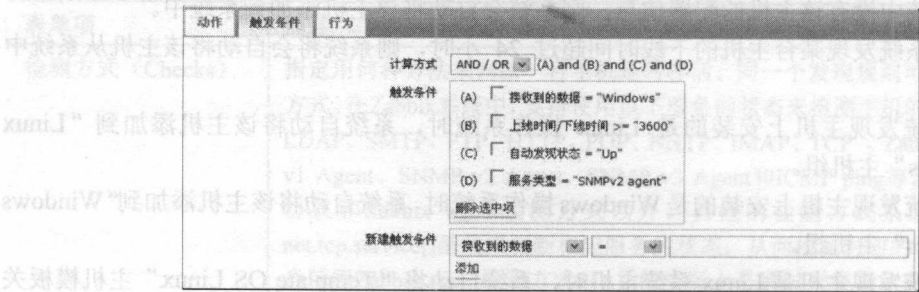


图 5-27 配置“自动添加 Windows 主机”动作触发条件表单页面截图

“自动添加 Windows 主机”动作“行为”选项卡上的配置如图 5-28 所示。

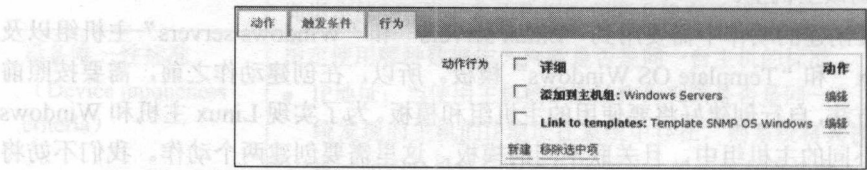


图 5-28 配置“自动添加 Windows 主机”动作行为表单页面截图

3. “自动删除主机”动作配置内容

“自动删除主机”动作“触发条件”选项卡上的配置如图 5-29 所示。

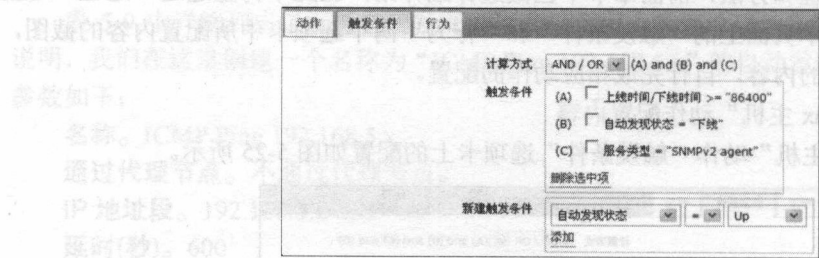


图 5-29 配置“自动删除主机”动作触发条件表单页面截图

“自动删除主机”动作“行为”选项卡上的配置如图 5-30 所示。

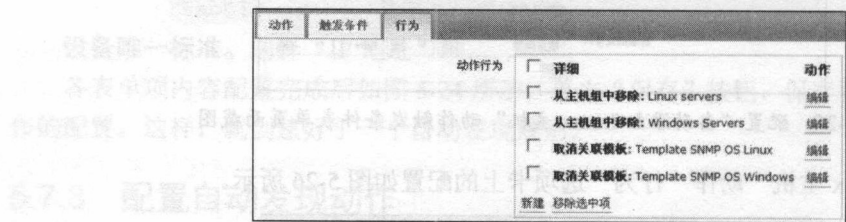


图 5-30 配置“自动删除主机”动作行为表单页面截图

5.8 Web 监控

虽然通过 Zabbix 系统所提供的“简单检查”方法，可以实现对 Web 系统的服务端口状态进行监控。但是在很多时候，仅对 Web 服务的端口状态进行监控是远远不够的。比如，很多时

候服务端口状态是正常的,但是因为某种原因导致应用程序出错了,这个时候实际上 Web 服务也是不可用的。同时,除了端口状态,很多时候,还需要对 Web 服务的性能数据进行监控。例如,服务的响应时间、数据的下载速度等。Zabbix 系统提供的“Web 监控(Web Monitoring)”功能,完全能满足我们对于这类监控的需求。接下来介绍什么是“Web 监控”,以及如何在 Zabbix 系统中配置“Web 监控”。

5.8.1 Web 监控介绍

Web 监控是一种对 Web 站点的可用状态、性能数据(例如下载速度、响应时间等)等进行数据采集和监控的功能。要使用 Zabbix 系统的“Web 监控”功能,需在编译安装 Zabbix 服务器端及其服务器代理端组件时,显式地使用“--with-libcurl”编译配置选项,以使 Zabbix 服务器端组件及其代理端组件支持通过 Curl 软件包所提供的函数库打开 Web 页面。

要想通过 Zabbix 系统对某个站点进行 Web 监控,则需要在 Zabbix 系统中配置针对这个站点的监控“场景(Web Scenarios)”和“步骤(Steps)”。“步骤”是由我们预先定义的,并且由 Zabbix 服务器(或其代理)按顺序周期性执行的 HTTP 请求。一个或多个步骤可以组成一个“场景”,而步骤里定义了 Web 监控的具体检查目标和方法。在场景和步骤层面上,可以采集到被监控对象上不同的监控数据。具体来说,通过场景和步骤可以采集到以下这些数据。

1. 针对 Web 场景,可以采集到的数据有:

- 在某个 Web 场景中,系统在执行完所有步骤后,计算出来的平均每秒下载速度,单位是“字节/秒”。
- 在某个 Web 场景中,系统执行失败或请求失败的步骤的总个数。
- 在某个 Web 场景中,系统在执行所有步骤的过程中,所接收到的最后一次出错信息。

2. 针对场景中的步骤,可以采集到的数据有:

- 每个步骤所对应的平均每 s 的下载速度,单位是字节每秒。
- 针对场景中的步骤,系统在执行步骤中所配置的请求时,所接收到的 Web 服务器响应时间。
- 针对场景中的步骤,系统在执行步骤中所配置的请求时,所接收到的 Web 服务器响应的 HTTP 代码。

在 Web 检查中,Zabbix 系统可以针对每个步骤,在被请求页面所返回的 HTML 代码中匹配步骤中所配置的指定字符串。通过这个方法,可以监控 Web 应用是否正常。同时,Zabbix 系统可以模拟用户登录及用户单击鼠标等动作。

Zabbix 系统中的 Web 监控,支持使用 HTTP 和 HTTPS 两种协议请求指定的 Web 页面,而且在整个场景的检查过程中,Zabbix 系统可以做到保留 Cookie 值。同时,它能跟踪页面的自动跳转。通俗地说,当在某个场景的某个步骤中所配置的 URL 是一个跳转页面所对应的 URL 时,Zabbix 系统在请求这个 URL 所对应的 Web 页面时,能很好地跟踪页面跳转。也就是说,Zabbix 系统能随着页面的自动跳转,自动检查跳转目标页面的状态。

当在系统中创建一个 Web 监控的场景和步骤时,Zabbix 系统会针对这个场景以及属于该场景下的每个步骤,自动创建多个监控项目。虽然,这些由系统自动创建的监控项目,我们在对应主机的“配置项目”列表页面上(在 Web 前端组件页面上依次选择菜单项“系统配置”→“主机”→“项目”,可以打开该页面)是查询不到的。但是,通过“状态统计”→“最新数据”

页面，我们仍然是可以查看这些监控项目所采集的最新数据的。而且，这些被 Zabbix 系统自动创建的监控项目，也可以像其他监控项目一样，被触发器、数据图等监控元素所引用。同时，Zabbix 系统针对 Web 监控所创建的监控项目，与其所对应的场景属于同一个分类。而且 Zabbix 系统在自动创建这些监控项目时，自动将这些监控项目的“保留历史数据”和“保留趋势数据”天数，分别配置为 30 天和 90 天。表 5-10 和表 5-11 分别列出了，Zabbix 系统针对 Web 监控的场景和步骤，自动创建的监控项目以及它们的含义。

表 5-10 Web 监控场景监控项目列表

监控项目	描述
场景的下载速度 (Download speed for scenario)	该监控项目用于采集，整个场景中所有步骤的平均下载速度，该下载速度是由 Zabbix 系统在执行场景中所有步骤时，根据所接收到的返回数据的大小除以执行这些步骤所消耗的总时间而计算出来的，单位是字节每秒 监控项目关键字: web.test.in[场景名,,bps] 信息类型: 数值 (浮点数)
场景的失败步骤数 (Failed step of scenario)	该监控项目用于采集，系统在执行整个场景中所有步骤时，执行失败的步骤个数。这里所说的执行失败，不仅仅是指 Zabbix 系统请求步骤中指定的 URL 不成功，比如服务器连接不上等，而且还包括步骤中所要求匹配的字符串在页面所返回的 HTML 源代码中没有被成功匹配到，或者步骤中所要求匹配的 HTTP 代码与 Web 服务器实际所返回的 HTTP 代码不匹配等。如果场景中所配置的所有步骤都执行成功，则该监控项的值为 0 监控项目关键字: web.test.fail[场景名] 信息类型: 数值 (无符号整型)
场景的最后一次错误信息 (Last error message of scenario)	该监控项目用于采集 Zabbix 系统在执行场景中所有步骤时，最后一次错误的错误信息。如果系统在执行一个场景中所有步骤的过程中，发生过多次错误，则该项目只接收最后一次错误所对应的错误信息。相应的，如果场景中的所有步骤被成功执行，则这个监控项目的数据不会被更新。举个例子，比如系统在 11:01 时执行场景中某个步骤发生“Couldn't connect to server”错误，而之后该场景中的所有步骤都被成功执行，即再没有其他步骤执行出错，则这个监控项目的监控数据就一直保持为“Couldn't connect to server”，直到该数据作为历史数据被清除为止 监控项目关键字: web.test.error[场景名] 信息类型: 文本

表 5-11 Web 监控步骤监控项目列表

监控项目	描述
步骤下载速度 (Download speed for step of scenario)	这个监控项目用于采集，Zabbix 系统请求步骤中所指定页面的下载速度，单位是字节每秒 监控项目关键字: web.test.in[场景名,步骤,bps] 信息类型: 数值 (浮点数)

续表

监控项目	描述
步骤的响应时间 (Response time for step)	这个监控项目用于采集, Zabbix 系统请求步骤中所指定的页面时, Web 服务器的响应时间, 单位是字节每秒。响应时间是从 Zabbix 系统开始请求指定的页面, 到该指定页面的内容全部下载完毕所用的时长 监控项目关键字: web.test.time[场景名, 步骤] 信息类型: 数值 (浮点数)
步骤的响应代码 (Response code for step of scenario)	这个监控项目用于采集, Zabbix 系统在请求步骤中所指定的页面时, Web 服务器所返回的 HTTP 响应代码 监控项目关键字: web.test.rspcode[场景名, 步骤] 信息类型: 数值 (无符号整型)

从表 5-9 和表 5-10 列出的监控项目可以看出, Zabbix 系统针对 Web 监控自动创建的监控项目, 其名称、监控项目关键字都有固定的格式。系统会针对每个场景, 自动在系统中创建表 5-9 所示的三个场景类监控项目。同样, 系统也会针对场景中的每个步骤, 自动在系统中创建如表 5-10 所示的三个步骤类监控项目, 可以引用这些监控项目创建触发器、数据图等。

如前面所介绍的, Zabbix 系统中所有的监控项目都必须依附于一台被监控主机, Web 监控也一样。但是, 在 Web 监控中, 我们在步骤中所指定的 URL 地址, 不一定非要是该 Web 监控项目所依附的被监控主机上所提供的服务地址, 它可以是其他主机上的 Web 页面。换句话说, 在某台主机上所配置的 Web 监控项目, 可以用来监控其他主机上的 Web 页面状态。

因为在 Web 监控的步骤中需要指定所监控页面的 URL, 而对于 Zabbix 系统来说, URL 所对应的主机名解析出来的 IP 地址是唯一的, 至少在一次检查的过程中是唯一的, 所以, Web 监控类项目不支持配置在模板和低级自动发现规则中。

提示: 1. 如果 Zabbix 服务器或其代理需要通过 HTTP 代理请求被监控页面, 则需要在运行 Zabbix 服务器(或其代理)端进程的用户(一般为 zabbix 用户)系统环境中添加上 http_proxy 环境变量。可以将形如 “export http_proxy=http://proxy_ip:proxy_port” 的内容添加到用户的 <USER_HOME>/.bash_profile 文件中, 以达到设置环境变量 http_proxy 的目的。其中, proxy_ip 为代理服务器的 IP 地址或主机名, proxy_port 为代理端口号。

2. 当 Zabbix 系统需要通过 HTTPS 协议请求被监控页面, 且需要通过 HTTPS 代理时, 需要在运行 Zabbix 服务器(或其代理)端进程的用户(一般为 zabbix 用户)的系统环境中添加上 HTTPS_PROXY 环境变量。环境变量的形式为: HTTPS_PROXY=http://proxy_ip:proxy_port。

5.8.2 Web 监控配置

上一节详细介绍了 Zabbix 系统中的 Web 监控功能。本节将来实际配置一个 Web 监控项目, 以用于采集和监控 Zabbix 系统的 Web 前端组件页面的性能数据和状态。

要创建 Web 监控项目, 依次选择 “系统配置” → “Web” 菜单项, 然后在 “配置 Web 监控” 页面的右边下拉菜单中选择要创建 Web 监控项目的主机, 最后单击页面上的 “新建 Web 场景” 按钮, 系统将打开如图 5-31 所示的 “配置 Web 监控” 表单页面。

从图 5-31 可以看出, 配置 Web 监控表单页面上有两个选项卡: “Web 场景” 和 “步骤”

选项卡。其中，“Web 场景”选项卡上各表单项用于配置 Web 场景信息。表 5-12 列出了该选项卡上部分需要稍加说明的表单项及它们的作用。

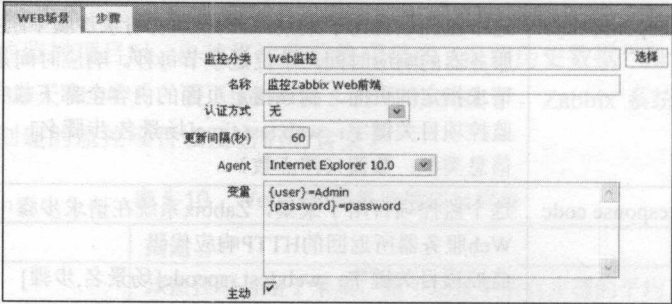


图 5-31 配置 Web 监控表单页面截图

表 5-12 配置Web场景表单项列表

表单项	描述
监控分类 (Application)	用于指定所要创建的Web场景属于哪个分类。如果该分类在系统中已经存在，则单击该表单项后面的“选择”按钮可以选择相应的分类。否则，如果在该表单项里输入了一个字符串，则系统在创建Web场景的同时，会自动创建一个以你所输入的内容为名称的监控分类，并将所创建的Web场景归属于该分类
认证方式 (Authentication)	用于配置认证方式，该表单项具有三个备选项： 无 (None) —— 不需要认证 基础验证 (Basic authentication) —— 使用HTTP协议中所定义的基础认证 (Basic authentication) 方法进行认证 NTLM验证 (NTLM authentication) —— NTLM是Windows提供的一个认证协议 需要说明的是，这里所说的认证方式，与我们经常接触到的网站登录是两回事。常见的一般网站登录，通常是网站程序结合数据库一起实现的，而这里所说的认证，是指通过HTTP协议或Windows等操作系统所提供的认证方法进行的认证。当选择一种认证方式后，页面上会多出两个表单项“用户”和“密码”。这两个表单项是用于填写用于认证的用户名和密码的
代理 (Agent)	用于选择Zabbix系统所模拟的浏览器种类及版本。Zabbix系统在请求场景步骤中所指定的页面时，可以模拟多个版本的浏览器。这个配置在某些时候比较重要，因为某些站点针对不同的浏览器，其特性和显示效果很不一样
变量 (Variables)	用于配置当前场景的步骤中将要使用的宏变量。它的配置格式是： {macro1}=value1 {macro2}=value2 即每一行定义一个宏变量。在步骤中，可以以{macro1}的形式使用在这里定义的宏变量。当Zabbix系统请求场景步骤中指定的URL页面时，会使用宏变量所指代的值替换对应的宏变量。 需要注意的是，在Web监控的步骤中，不可以使用我们在前面章节中介绍的全局宏变量、模板级宏变量和主机层面的宏变量，而只能使用在场景中定义的宏变量。而且，在场景中定义宏变量，也不像在其地方定义宏变量所要求的那样，必须使用全大写的字母来定义宏变量

这里将要创建的 Web 监控项目“监控 Zabbix Web 前端”，在“场景”选项卡上各表单项的配置如图 5-31 所示。其中，在这个场景中，创建了两个宏变量：{user}和{password}，以用于在后面将要创建的步骤中登录 Zabbix 系统。

接下来单击“步骤”选项卡，并单击“步骤”选项卡页面上的“添加”超链接，系统将打开如图 5-32 所示的配置步骤的表单页面。

图 5-32 配置步骤表单页面截图

配置步骤的表单内容比较简单，表 5-13 列出了该表单页面上需要稍加说明的表单项及其作用和含义。

表 5-13 配置步骤表单项列表

表单项	描述
URL	用于指定 Zabbix 系统所要请求的页面的 URL。例如，http://zabbix.domain.com/zabbix/index.php等。在这个 URL 中，可以将场景中定义的宏变量作为 URL 参数来使用
Post	用于指定 Zabbix 系统模拟浏览器所提交的表单数据。格式可以写成 name={user}&password={password}&enter=Sign in 形式。其中，{user}和 {password}即为在场景中定义的宏变量，当 Zabbix 系统请求指定的页面时，它首先会用宏变量所指代的值替换掉这里的宏变量，然后将替换后的数据提交给 Web 服务器。但是，在这个表单项里所配置的内容，在 Zabbix 系统提交给 Web 服务器时不会进行 URL 编码
超时（Timeout）	用于指定 Zabbix 系统处理步骤中所指定的 URL 请求时所用的最大超时时间。需要注意的是，这个超时时间不是指 Zabbix 系统从连接 Web 服务器到处理完指定请求所用的总的最大超时时间。而是指，Zabbix 系统连接 Web 服务器或处理完指定请求所用的最大超时时间
匹配的字符串 (Required string)	用于指定需要在 Web 服务器返回的 HTML 代码中匹配的字符串。如果该表单项中指定的字符串，在 Web 服务器返回的 HTML 代码中被匹配到，则表示指定 URL 页面正常；否则，Zabbix 系统视请求指定页面失败。而如果这个表单项里不输入任何内容，则 Zabbix 系统不进行这种匹配
匹配的状态代码 (Required status codes)	该表单项中所输入的内容，将会被 Zabbix 系统用来与 Web 服务器所响应的 HTTP 代码进行匹配。如果 Web 服务器所响应的 HTTP 代码不在该表单项所指定的列表内，则 Zabbix 系统视该步骤检查失败。因此，该表单项的内容可以是一个列表，例如 200、201、210-299 等。如果这个表单项里所输入的内容为空，则 Zabbix 系统不进行上述的响应代码匹配

接下来，结合表 5-12 中列出的配置步骤表单页面中各表单项的含义和作用，我们在“监控 Zabbix Web 前端”场景中配置如下 4 个“步骤”。这里仅列出配置这 4 个“步骤”的各表单项内容，不一一说明具体配置的过程，请读者自行完成这 4 个“步骤”的配置。

步骤一

名称：首页面。

URL: `http://zabbix.domain.com/zabbix/index.php`。
Required string: Zabbix SIA。
Required status codes: 200。

步骤二

名称：登录页面。

URL: `http://zabbix.domain.com/zabbix/index.php`。
Post: `name={user}&password={password}&enter=Sign in`。
Required status codes: 200。

步骤三

名称：登录检查页面。

URL: `http://zabbix.domain.com/zabbix/index.php`。
Required string: 配置。
Required status codes: 200。

步骤四

名称：退出页面。

URL: `http://zabbix.domain.com/zabbix/index.php?reconnect=1`。
Required status codes: 200。

当完成上述 4 个步骤的配置后，它们的列表信息如图 5-33 所示。

步骤	名称	超时	URL	需求	状态代码	动作
# 1:	首页面	15 秒	<code>http://zabbix.domain.com/zabbix/index.php</code>	Zabbix SIA	200	移除
# 2:	登录页面	15 秒	<code>http://zabbix.domain.com/zabbix/index.php</code>		200	移除
# 3:	登录检查页面	15 秒	<code>http://zabbix.domain.com/zabbix/index.php</code>	配置	200	移除
# 4:	退出页面	15 秒	<code>http://zabbix.domain.com/zabbix/index.php?reconnect=1</code>		200	移除
添加						

图 5-33 步骤信息列表

前面在“监控 Zabbix Web 前端”场景中配置了 4 个步骤。其中，步骤一用于检查用户登录表单页面是否正常，检查方法是匹配该页面的 HTML 源代码中是否包含有“Zabbix SIA”字符串，及 Web 服务器所响应的 HTTP 代码是否是 200；步骤二用于模拟用户登录，其中宏变量 {user} 和 {password} 是在场景中定义的；步骤三则用于检查模拟登录是否成功，如果模拟登录成功，则在 Web 服务器所返回的 HTML 源代码中包含有字符串“配置”；而步骤四则用于模拟用户退出系统，类似于用户用鼠标单击 Zabbix 系统 Web 前端页面上的“退出”超链接。当完成上述场景和步骤的配置后，单击页面上的“保存”按钮，保存我们对场景和步骤所做的配置。过几分钟后，依次选择菜单项“状态统计”→“Web”，系统将打开“Web 监控状态页面”，从该页面上的“组”和“主机”下拉菜单中，选择我们刚刚在其上配置 Web 监控项目所对应的主

机，单击“Zabbix Web 前端”超链接，就可以看到如图 5-34 和图 5-35 所示的数据图。

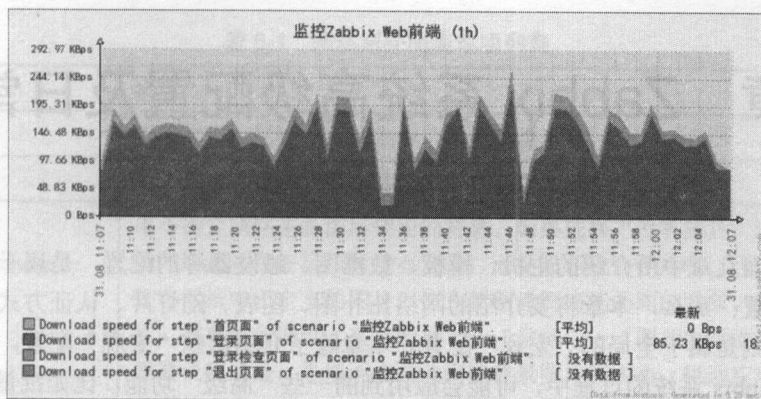


图 5-34 Zabbix Web 前端页面下载速率数据图

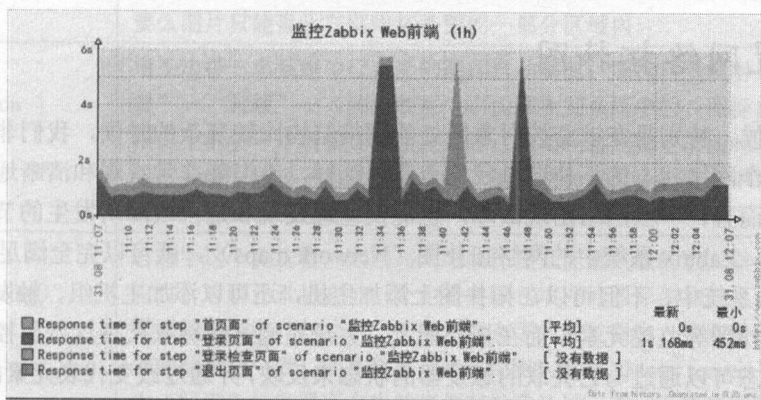


图 5-35 Zabbix Web 前端服务器响应数据图

5.9 本章小结

本章详细介绍了模板、正则表达式、低级自动发现等方面的内容，并进一步介绍了触发器表达式、触发器等级、动作的触发条件和行为等内容。并且，在本章实际动手创建了多个项目样板、触发器样板和数据图样板等。如果将在本章所创建的模板关联到某台被监控主机上，则就可以实现对该台被监控主机实时采集监控数据、生成监控项目的数据图以及实现对被监控对象的实时监控与报警等功能。通过本章的学习，我们就可以使用 Zabbix 系统来完成日常绝大部分系统监控工作了。

学习了本章，相信读者对于 Zabbix 系统功能之强大、配置之灵活，应该有一定的认识和感受了吧。而且，应该说，系统监控是一项实践性很强的工作，它需要我们有广阔的知识面和较强的动手能力。因此，如果要灵活、高效地使用 Zabbix 系统所提供的各种功能，我们还需要持续不断地对 Zabbix 系统进行研究和测试，从而积累丰富的经验，并以此为基础，持续对监控方法、配置方法进行优化和改进。

第 6 章 Zabbix 系统高级配置及日常管理

如果说前面几章中所介绍的主机、模板、数据图、触发器等配置，是属于针对监控元素的“微观”配置；那么，本章将要介绍的网络拓扑图、图表、幻灯片、认证方式、用户和用户组等的配置，则是属于全局的“宏观”配置。本章除了介绍这些“宏观”配置，还会介绍一些在日常维护 Zabbix 系统的过程中，可能会应用到的一些“高级”功能，比如批量更新、配置的导出与导入、全局搜索等。

6.1 配置网络拓扑图

在很多时候，特别是在被监控对象所处的网络结构比较复杂的时候，我们非常希望有一张能反映当前整个网络状态的拓扑图供我们查看。这样，不但能非常直观和清晰地监控整个网络的当前状态，而且，一旦网络出现故障，也能快速发现和定位故障所发生的节点，从而大大提高工作效率。Zabbix 系统中的网络拓扑图（Network maps），就可以完全满足这些要求。

在 Zabbix 系统中，不但可以在拓扑图上添加主机，还可以添加主机组、触发器、图片甚至是另外一个拓扑图等监控元素。而在拓扑图上，系统是通过图标来代表这些监控元素的。拓扑图上元素的状态可以通过与它关联的触发器的状态来反映，并通过改变代表元素的图标的大小、样式、颜色呈现在拓扑图上。

6.1.1 定义网络拓扑图

要创建一张网络拓扑图，首先需要在系统中创建一个网络拓扑图的定义，然后再在这个定义好的网络拓扑图上添加和配置相关元素。而要定义一张网络拓扑图，需依次选择“系统配置”→“拓扑图”菜单项，并在“配置网络拓扑图”页面上，单击“新建拓扑图”按钮，系统将打开如图 6-1 所示的定义网络拓扑图的表单页面。

名称	URL	元素
		主机

图 6-1 定义网络拓扑图表单页面截图

图 6-1 所示的表单页面中部分表单项的作用和含义如表 6-1 所示。

表 6-1 网络拓扑图表单项列表

表单项	描述
宽 (Width)	用于定义网络拓扑图的宽度，单位是像素
高 (Height)	用于定义网络拓扑图的高度，单位是像素
背景图片 (Background image)	用于指定网络拓扑图所使用的背景。如果选择“没有图片”选项，则所定义的网络拓扑图在显示时没有背景，而是显示成白色底色。否则，可以选择一张在系统中已定义的图片，作为新创建网络拓扑图的背景。被选择的背景图片，需要提前通过“高级配置”→“常规”→“图片”页面添加到系统中。需要说明的是，系统不会自动调整背景图片的大小，以适应所定义的网络拓扑图的大小。所以，当要为一张网络拓扑图选择背景图片时，所选择的背景图片大小最好与网络拓扑图定义的大小相一致，否则，要么图片显示不完整，要么图片只能显示在网络拓扑图的一部分区域内
自动图标映射 (Automatic icon mapping)	用于选择一个系统中已定义的图标自动映射，图标自动映射是通过“高级配置”→“常规”→“图标映射”页面添加到系统中的。图标自动映射，可以根据主机资产中所指定的字段内容，自动映射到指定的图标。如果在一张网络拓扑图中选择了自动图标映射，则系统会根据主机资产中相应的字段内容显示主机的相应图标
图标高亮 (Icon highlighting)	若选中这个复选框，则当与网络拓扑图上元素关联的触发器处于“问题”状态时，对应元素将会被一个由特定颜色填充的圆形图加亮。而填充这个加亮圆形图的颜色，与处于“问题”状态下的触发器中，级别最高的触发器所配置的颜色相一致。加亮效果，请参看图3-38中“Linux servers”主机组。如果由“问题”触发器所产生的事件都被确认过，则这个加亮圆形图的圆周将显示成绿色的细线。如果网络拓扑中的元素处于禁用或维护状态，则加亮图形将会是灰色或橙色的矩形图，如图3-38中主机“192.168.5.139”所显示的效果
当触发器状态改变时标记元素 (Mark elements on trigger status change)	如果选中该复选框，则当有与网络拓扑图中元素相关的触发器的状态发生改变时，包括从“问题”状态转变为“正常”状态，和从“正常”状态转变为“问题”状态，则系统会在与该触发器有关联的元素图标周围用红色的三角形来标识，如图3-38中“Linux servers”和主机“192.168.5.139”所显示的效果。这种显示效果会在触发器状态改变之后30分内持续地显示在相关的元素图标上
Expand single problem	这个复选框用于控制，当有与网络拓扑图中元素（包括主机、主机组或其他拓扑图等）相关联的触发器处于“问题”状态时，系统是显示该触发器的名称还是显示处于“问题”状态下的触发器数量。当选中这个复选框时，系统会显示触发器的名称；否则，系统就会显示触发器的数量
高级标签 (Advanced labels)	当选中这个复选框时，系统会在当前表单页面上自动增加“主机组标签类型”、“主机标签类型”、“触发器标签类型”和“拓扑图标签类型”等表单项，它们是用于指定不同种类的元素所使用的标签类型
图标标签类型 (Icon label type)	用于指定系统使用哪类信息作为图标的标签显示在网络拓扑图上。可用的选项有：标签、IP地址、元素的名称、状态以及不显示标签等

续表

表单项	描述
图标标签位置 (Icon label location)	用于指定在图标的什么位置显示元素的标签，可选项有：底部 (Bottom)、左边 (Left)、右边 (Right) 和上部 (Top)
问题显示 (Problem display)	用于配置用何种方式显示元素的“问题”个数，可选项有： 全部——统计并显示与元素相关联的“问题”触发器个数 分隔的——分别统计与元素相关的未确认事件数和总事件数，并分别显示仅未确认的——仅统计和显示与元素相关的未确认事件数
URLs	用于针对不同类型的元素，配置额外的URL连接。这里所配置的URL，在用户通过“状态统计”→“拓扑图”菜单项查看拓扑图时，可以显示在用户的右键弹出菜单中。当用户单击对应的菜单项时，可以打开指定的页面。在这里定义的URL是针对元素类型的，所以它会在所有同一类型元素的右键菜单中出现。这个表单项可以支持 {MAP.ID}、{HOSTGROUP.ID}、{HOST.ID} 和 {TRIGGER.ID} 等宏变量

按照表 6-1 的说明，定义一个名叫“全国各地机房拓扑图”的拓扑图，各表单项的内容如图 6-1 所示。

6.1.2 编辑网络拓扑图元素

前面在系统中定义了一张网络拓扑图。接下来，需要在这张网络拓扑图上添加需要显示的元素，并组织好它们之间的关系。这样，所创建的拓扑图才有实际效果和意义。

依次选择“系统配置”→“拓扑图”菜单项，并在“配置网络拓扑图”页面上，找到刚刚创建的“全国各地机房拓扑图”，并单击名称上的超链接，系统将打开如图 6-2 所示的配置网络拓扑图元素的页面。

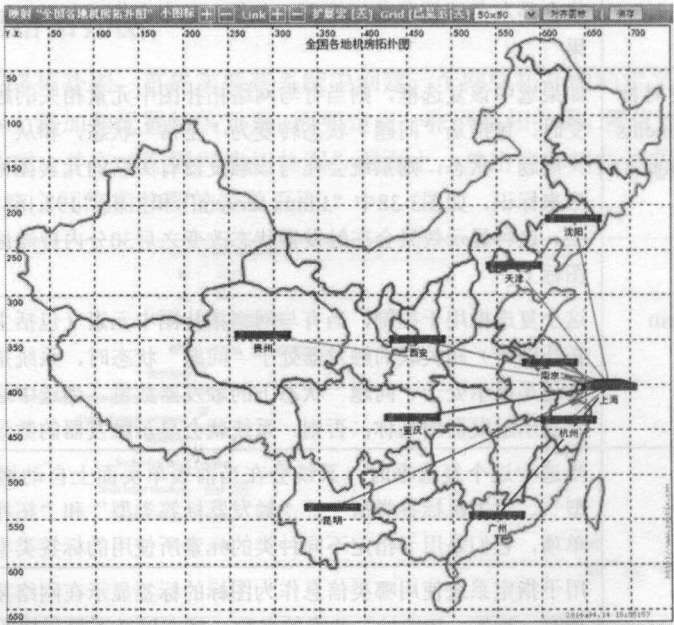






图 6-2 配置网络拓扑图元素页面截图

如图 6-2 所示，单击标题栏上“小图标”后面上的超链接，就可以在拓扑图上添加新的元素。新添加的元素最初被放置在拓扑图的右上角，用鼠标选中它，就可以将它拖放到我们希望的位置上。如果在拓扑图上选中一个元素，然后单击标题栏上“小图标”后面符号上的超链接，就可以将所选定的元素从拓扑图上删除。相应的，标题栏上“Link”后面的和图标分别用于添加和删除两个元素之间的连接线。如果打开“扩展宏”开关，则当元素标签中包含宏变量时，系统将以对应宏变量所指代的值替换宏变量；而如果关闭“扩展宏”开关，则当元素标签中包含宏变量时，系统直接在拓扑图上显示该宏变量而不是它所指代的值。与网格(Grid)对应的有两组开关，它们分别是用于控制是否显示网格线的“隐藏/已显示”开关和用于控制元素是否对齐网格的“开/关”。当网格对齐开关打开时，如果向拓扑图上放置元素，则系统会自动将该元素与网格对齐。

当在拓扑图编辑区域选中一个元素时，系统会弹出如图 6-3 所示的配置元素属性的表单页面。通过这个表单页面，可以对所选定元素的属性进行编辑，包括修改元素的类型、默认图标、关联元素等。当在拓扑图编辑区域选中元素时按住 Ctrl 键不放，则可以一次选择多个元素。如果一次选中了多个元素，则系统会显示出“批量更新元素”表单页面。通过这个表单页面，可以一次对多个元素的某个或某些属性进行修改和更新。批量修改元素属性和修改单个元素属性的操作方法和过程差别不大。并且，我们在后续章节中还将详细介绍 Zabbix 系统中的批量更新功能。所以，这里我们不单独介绍如何批量更新元素的属性，仅介绍如何修改单个元素的属性。

编辑拓扑图元素

类型

主机

Label

南京监控服务器

Label location

底部

主机

192.168.5.139

选择

图标

Automatic icon selection

默认

Zabbix_server_3D_(64)

维护模式

Zabbix_server_3D_(64)

Problem

Zabbix_server_3D_(64)

禁用

Zabbix_server_3D_(64)

Coordinates

X: 618

Y: 344

URLs

名称

URL

删除

添加

应用

移除

关闭

编辑元素链接

元素类型	元素名称	Link indicators
编辑 主机	天津监控服务器	

图 6-3 配置元素属性表单页面截图


图 6-3 所示的配置元素属性表单页面上各表单项的作用和含义如表 6-2 所示。

表 6-2 配置元素属性表单项列表

表单项	描述
类型 (Type)	用于选择元素类型，该表单项有如下备选项： 主机 (Host) —— 如果选择这个选项，则对应元素代表与指定主机相关的所有触发器的状态 拓扑图 (Map) —— 如果选择这个选项，则对应元素代表拓扑图上所有元素的状态 触发器 (Trigger) —— 如果选择这个选项，则对应元素代表指定的某个触发器的状态 主机组 (Host group) —— 如果选择这个选项，则对应的元素表示与属于所选主机组中的所有主机有关的所有触发器的状态 图片 (Image) —— 如果选择这个选项，则对应元素仅仅是一张图片，不与任何资源相关联
标签 (Label)	用于配置元素的标签。元素的标签可以是任何字符串，且可以是多行字符串；同时在元素的标签中可以使用系统中定义的宏变量
标签位置 (Label location)	用于指定在网络拓扑图上，元素的标签显示在元素图标的位置。可选项有“默认 (Default)”、“底部 (Bottom)”、“左侧 (Left)”、“右侧 (Right)”和“顶部 (Top)”等
主机 (Host)	如果元素的类型是主机，则通过这个表单项指定对应元素所关联的主机
拓扑图 (Map)	如果元素的类型是拓扑图，则通过这个表单项指定对应元素所关联的拓扑图
触发器 (Trigger)	如果元素的类型是触发器，则通过这个表单项指定对应元素所关联的触发器
主机组 (Host group)	如果元素的类型是主机组，则通过这个表单项指定对应元素所关联的主机组
自动选择图标 (Automatic icon selection)	如果选中这个复选框，则系统在显示对应元素不同状态下的图标时，会使用在定义拓扑图属性时所指定的自动图标映射进行显示，否则会使用“图标”表单项中所指定的图标进行显示
图标表单项	用于指定元素在默认状态、问题状态、维护状态和禁用状态下分别用哪个图标显示在拓扑图上
横轴坐标 (Coordinate X)	用于指定元素在拓扑图上的X轴坐标
纵轴坐标 (Coordinate Y)	用于指定元素在拓扑图上的Y轴坐标
URLs	用于配置与元素相关联的URL地址。当用户通过“状态统计”→“拓扑图”菜单项打开指定的拓扑图时，在这里所配置的URL连接，将会显示在对应元素的右键菜单上。当在配置元素属性时针对某个元素配置了URL连接，而且在定义拓扑图时针对该元素所对应的类型也配置了URL连接，则在显示拓扑图时，这两个地方所配置的URL连接都将会显示在右键菜单上，且系统将合并相同项后再进行显示。所配置的URL连接信息中可以包含 {MAP.ID}、{HOSTGROUP.ID}、{HOST.ID}及{TRIGGER.ID}等宏变量

这里按照上述介绍的方法，并结合表 6-2 所列的配置拓扑图元素属性的表单页面中各表单项的含义和作用，在我们之前所定义的拓扑图——全国各地机房拓扑图上，添加如图 6-2 所示的各种拓扑图元素。

当在拓扑图上添加好所需要的各种拓扑图元素后，接下来就需要在它们之间创建连接线，以将它们之间的关系清晰地表现在拓扑图上。创建连接线的方法是，首先在拓扑图的编辑区域

中，选中需要创建连接线的两个元素，然后单击标题栏上“Link”后面图标上的超链接。当在两个元素之间创建了连接线后，系统会在配置元素属性的表单页面上增加一个如图 6-4 所示的“编辑元素链接”的表单域。

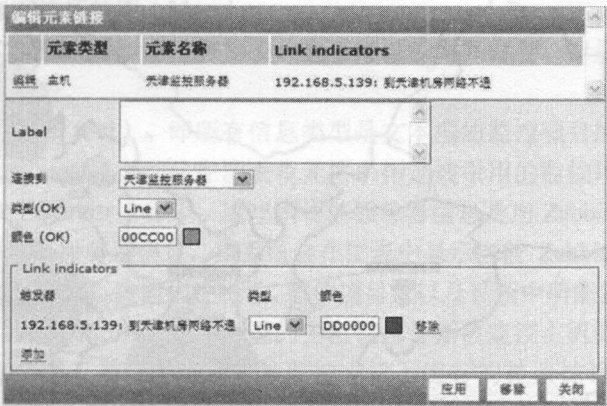


图 6-4 配置元素连接属性表单页面截图

图 6-4 所示的“编辑元素链接”表单域中各表单项的作用和含义如表 6-3 所示。

表 6-3 配置元素连接线属性表单项列表

表单项	描述
标签 (Label)	用于指定连接线的标签，该标签会显示在相应的连接线上。连接线标签内容支持使用宏变量
连接到 (Connect to)	用于指定该当前元素与哪个元素相连
默认状态下连接线类型 [Type (OK)]	用于指定连接线的类型。系统支持的连接线类型有细线 (Line)、粗线 (Bold Line)、点画线 (Dot) 和虚线 (Dashed line)
默认状态下连接线颜色 [Colour (OK)]	用于指定正常状态下连接线的颜色，可以通过调色板来选择
连接线指示器 (Link indicators)	用于指定与连接线关联的触发器及当这些被关联的触发器处于“问题”状态时，连接线的颜色和样式。一条连接线可以与多个触发器进行关联，当这些被关联的触发器处于“问题”状态时，则它们将影响连接线显示的颜色和样式

从表 6-3 中可以看出，连接两个元素之间的连接线在正常状态下，显示的颜色和样式是由表单项“默认状态下连接线类型”和“默认状态下连接颜色”来配置的。而当连接线关联了触发器时，则被关联触发器的状态会反映在连接线的颜色和样式上。当一条连接线关联了某个触发器时，系统将会以与这个触发器相关的颜色和样式来显示相应的连接线。而当一条连接线关联了多个触发器，且同时有多个触发器处于“问题”状态下时，则系统将以这些处于“问题”状态下的触发器中，级别最高的触发器所指定的颜色和样式来显示对应的连接线。而如果处于“问题”状态下的触发器中，具有级别相同的多个触发器，则系统将以在系统数据库中 ID 值最小的那个触发器所指定的颜色和样式来显示对应的连接线。

当根据表 6-3 中的说明，在“全国各地机房拓扑图”上配置好各元素之间的连接线后，等几分钟，依次选择菜单项“状态统计”→“拓扑图”，并从页面右边的下拉菜单里选择刚刚配置的拓扑图，就可以看到拓扑图的显示效果了，如图 6-5 所示。

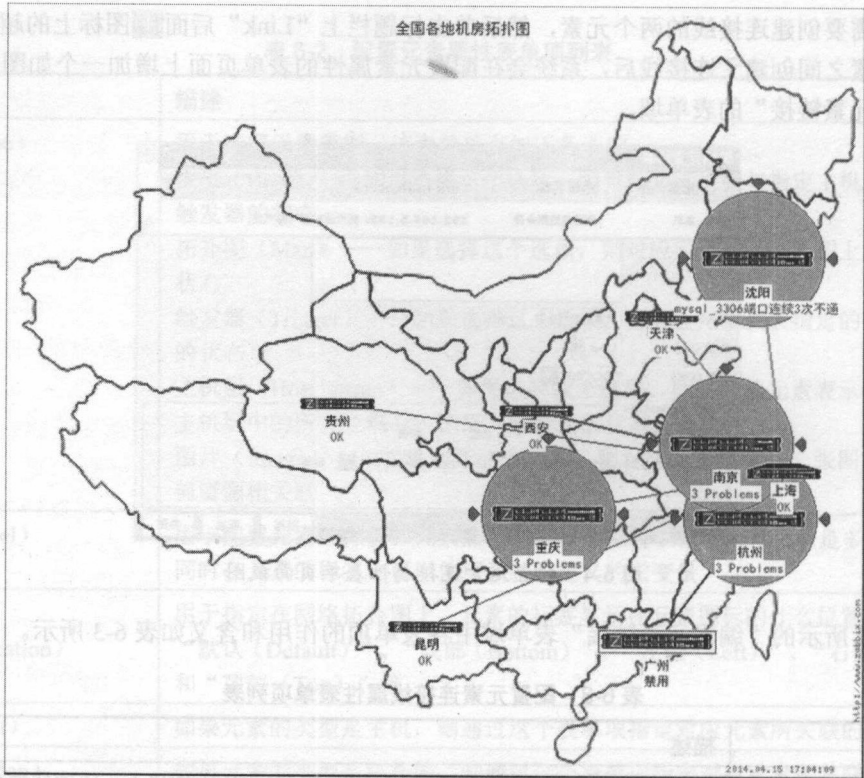


图 6-5 全国各地机房拓扑图

注意：当在拓扑图编辑区中对拓扑图元素做了修改和调整，系统不会自动保存修改后的信息，需要我们手动单击标题栏上的“保存”按钮，系统才会将修改后的信息保存到系统数据库中。所以，完成拓扑图上元素的编辑后，需要及时地手动将修改后的信息保存起来；否则，之前所做的所有修改在浏览器窗口被关闭时，就将全部丢失。

6.2 配置图表和幻灯片

图表 (screens)，从这个名字可以看出，在图表页面上系统既可以显示图（即数据图等），也可以显示表格。在 Zabbix 系统的图表页面中，可以将前面章节中所介绍的几乎所有种类的监控元素都放置到其页面上。例如，系统自动为我们所生成的简单数据图、我们自己定义的数据图、事件信息汇总、Zabbix 系统状态信息汇总、属于各主机组的触发器等监控元素，都可以将它们放置在一张图表页面中。而幻灯片用来按照一定的时间间隔来播放图表。本节将介绍如何配置图表和幻灯片。

6.2.1 配置图表

从本质上来说，Zabbix 系统中的图表其实就是表格。只是对于这类表格，可以自定义它的行列数以及它的每个单元格里所显示的内容。具体来说，可以放置在图表单元格的监控元素，可以包含以下所列的这些内容：

- ❑ 单数据图 (Simple graphs)。也就是针对信息类型是数值型的监控项目, 系统自动为我们所创建的数据图。
- ❑ 用户自定义的数据图 (Graph)。即通过手工所创建的数据图。
- ❑ 系统中已存在的拓扑图 (Map)。
- ❑ 其他图表 (Screen)。即一张图表, 它不可以引用其自身, 但是可以引用在系统中已定义的其他图表。
- ❑ 文本信息 (Plain text)。即所有信息类型是文本型的监控项目所采集的历史数据, 图表都可以引用, 引用方法是在图表单元格中指定被引用的监控项目。
- ❑ 服务器信息 (Server info)。这里所说的服务器信息是指 Zabbix 服务器的信息。如果在图表里引用了服务器信息, 则系统会在图表中显示当前 Zabbix 服务器最近一次更新时间、在线用户列表、被监控主机数、监控项目数以及系统中所配置的触发器数等信息。
- ❑ 主机信息 (Host info)。这里所说的主机信息是指被监控主机的状态信息。实际上, 系统统计的是属于指定主机组的被监控主机上运行的被监控设备代理服务的状态信息。分为“可用”数, 是指 Zabbix 服务器所检测到的, 被监控设备代理服务处于正常状态的主机数; “不可用”数, 是指被监控主机上已经配置了被监控设备代理数据采集接口, 但是 Zabbix 服务器检查到被监控设备代理服务的状态为不可用状态的被监控主机数; 而“未知”数则是指, 没有配置被监控设备代理数据采集接口的被监控主机数。所以, 这里所说的主机可用和不可用, 并不代表实际上真的有对应数量的主机处于存活和不存活状态, 而仅是指在配置了被监控设备代理数据采集接口的主机中, 有多少数量的主机上所运行的被监控设备代理服务的状态为可用和不可用。
- ❑ 主机组触发器状态 (Status of hostgroup triggers)。用于显示属于指定主机组的主机上的, 且处于“问题”状态下的触发器的列表信息。
- ❑ 主机的触发器状态 (Status of host triggers)。用于显示与指定主机相关的触发器状态列表, 但是只显示处于“问题”状态的触发器状态列表。
- ❑ 系统状态 (System status)。用于显示 Zabbix 系统的各种状态统计信息, 其内容包括各主机组中处于各种触发器级别状态下的主机数据统计信息。
- ❑ 数据总览 (Data overview)。用于显示指定主机组中各主机所采集的最新数据, 其所显示的内容与依次选择菜单项“状态统计”→“数据总览”时所显示的内容是类似的。
- ❑ 时钟 (Clock)。如果选择这种类型的资源, 则系统在图表的指定单元格中显示一个数字时钟, 时钟所显示的时间可以是本地时间也可以是 Zabbix 服务器上的系统时间, 还可以是指定服务器上的系统时间。
- ❑ 事件历史记录 (History of events)。显示系统中事件的历史记录数据。
- ❑ 动作历史记录 (History of actions)。显示系统中所记录的已经被触发的动作历史数据。
- ❑ URL。用于显示指定 URL 所对应的 Web 页面内容。

配置图表的方法非常简单明了。首先, 需要创建一个图表的定义。方法是, 依次选择菜单项“系统配置”→“图表”, 单击页面右上部的“新建图表”按钮, 系统将打开如图 6-6 所示的定义图表表单页面。

在图 6-6 所示的定义图表表单页面上, 只需输入需要创建图表的名称、初始行数和列数。其实, 在这个表单上, 唯一必须输入的内容只有“名称”, 其他表单项的内容都可以保持默认值。因为, 在向图表中添加元素时, 可以实时调整所需要的行数和列数。我们这里定义一个名

称为“my_screens”的图表。

图表

名称

my_screens

列数

2

函数

2

图 6-6 定义图表表单页面截图

在系统中创建好图表的定义后，在“配置图表”列表页面上，找到刚刚定义的名称为“my_screens”的图表，并单击其名称上的超链接，系统将打开配置图表元素的页面，如图 6-7 所示。

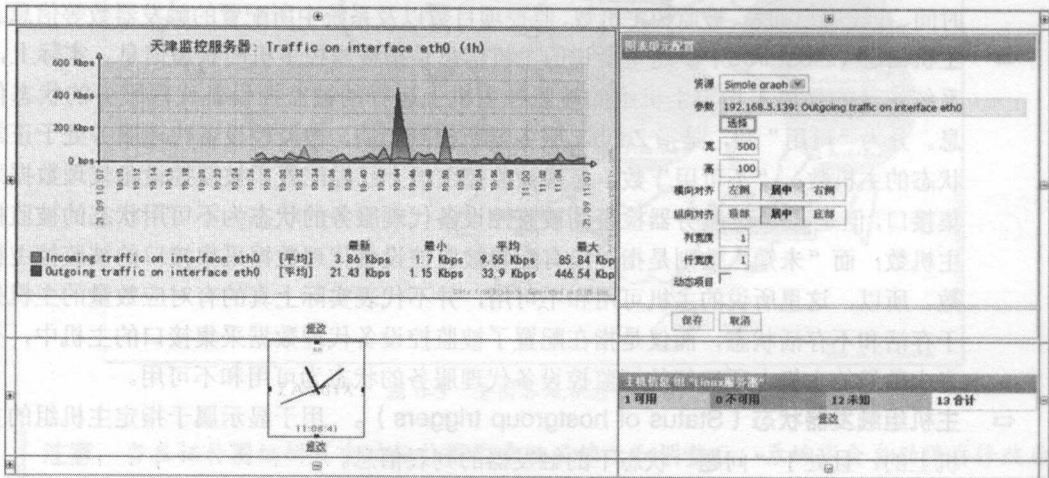


图 6-7 配置图表资源表单页面截图

从图 6-7 可以看出，整个图表其实就是一张表格。在这张表格中，其外围的单元格（即这张表格四周最边上的单元格）中显示的是“+”或“-”符号。我们这里将这些单元格称为功能区，而功能区所围成的内部单元格，我们称之为数据区。单击功能区上部（也就是这张表格的最上边一行）中的任何一个“+”号上的超链接，可以在这个表格中增加一列；单击功能区最下边（也就是这张表格的最下边一行）中的任何一个“-”号上的超链接，则可以删除这个表格中一列。类似的，单击表格左则“+”号上的超链接，则可以在这个表格中增加一行；而单击表格右侧“-”号上的超链接，则可以删除一行。

在数据区的每个单元格中都有一个“修改”超链接，单击这个超链接，系统将会在对单元格内显示一个表单域。通过这个表单域，可以对对应单元格内要显示的资源信息进行修改和配置，如图 6-7 中数据区的右边上面的单元格中所显示的内容。

配置图表资源的表单比较简单，主要包括“资源”：用于指定资源的类型；“高”：指定用于显示资源的区域的高度；“宽”：指定用于显示资源的区域的宽度；“模向对齐”方式、“纵向对齐”方式以及“动态项目”等。其中，需要稍加说明的表单项是“动态项目”这个表单项。当在配置图表资源时，对于某些类型的图表资源，系统会在对应表单域里显示一个叫做“动态项目”的表单项。当尝试选或不选中这个复选框时，系统并不会立即让我们看出它们有什么效果上的差异。然而，当依次选择菜单项“状态统计”→“图表”来浏览对应图表时，差

异就会立即体现出来。如果在配置图表资源时，选中“动态项目”复选框，则在浏览对应图表时，系统会在相应的页面上显示一个下拉菜单，以用于选择在图表中显示哪个主机组或主机的动态资源。这样，我们所定义的图表就有点类似于一个图表模板。例如，在配置图 6-7 中“监控服务器：网卡流量（发送）eth0(1h)”资源时选中了“动态项目”复选框，则该资源所在的单元格不但可以用来显示“监控服务器”这台主机 eth0 网卡的发送流量数据图，而且也可以用于显示其他主机上 eth0 网卡的发送流量数据图。其方法就是在显示图表时，通过前面所说的下拉菜单选择图表所需要显示资源所在的主机。然而，如果在一张图表中既有动态项目还有非动态项目，则当通过上述的下拉菜单来改变所需要显示的资源所在的主机时，非动态项目所对应的单元格里所显示的内容不会改变。

如前文所述，在图表中，只有某些类型的资源可以被设置成动态项目。支持被设置成动态项目的资源种类有：数据图（包括简单数据图、自定义数据图）和文本信息等。

依照前文的介绍，我们在“my_screens”图表上添加上一个 eth0 网卡发送流量数据图、一个 eth0 网卡接收流量数据图、一个时钟和一个主机信息。当完成上述这些资源的配置后，依次选择菜单项“状态统计”→“图表”，就可以查看我们所创建的图表了，其效果如图 6-8 所示。

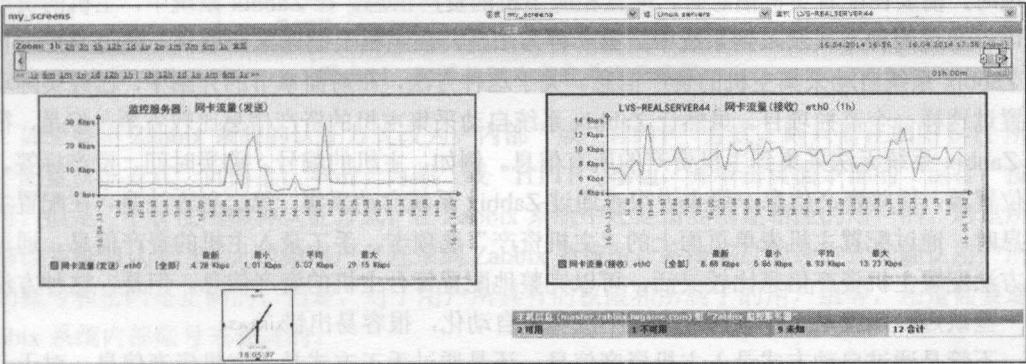


图 6-8 my_screens 图表效果图

6.2.2 配置幻灯片

相信大家对于幻灯片（Slide）的概念并不陌生，在 Zabbix 系统中，幻灯片是用于播放图表列表的。一组幻灯片中可以包含多张图表，系统在播放幻灯片时，会根据幻灯片中所配置图表的顺序和延时，循环播放这些图表页面，一张幻灯片播放一个图表页面。因此，在创建幻灯片之前，需要先在系统中创建好需要的图表。

配置幻灯片的方法很简单，依次选择菜单项“系统配置”→“幻灯片播放”，然后单击页面上的“新建幻灯片播放”按钮，系统将打开如图 6-9 所示的“配置幻灯片播放”表单页面。

名称	my		
默认延迟(秒)	10		
幻灯片	图表	延迟	动作
\$ 1	运营网络公网流量汇总	默认	移除
\$ 2	my_screens	默认	移除
添加			

图 6-9 配置幻灯片播放表单页面截图

如图 6-9 所示，配置幻灯片表单页面上的内容比较简单，只需输入幻灯片的名称、默认延时（秒）等内容，并单击“幻灯片”表单域中的“添加”超链接，向幻灯片中添加需要的图表即可。“幻灯片”表单域中显示的图表的顺序，即为系统在播放对应幻灯片时的播放顺序。如果某个图表项被指定了延时时间，系统在播放图表时，该图表每一轮显示的时长即为“延时”表单项中所指定的时间。例如，图 6-9 中我们所配置的幻灯片，在播放时的顺序和延时为：系统打开幻灯片并显示“my_screens”图表，显示时间是 5ns，接下来显示“运营网络公网流量汇总”图表，显示时间是 10s，再接下来进行下一轮显示，如此循环播放。

按照图 6-9 所示的配置信息配置好“my”幻灯片后，依次选择菜单项“状态统计”→“图表”，并在“配置图表”页面上的下拉菜单中选择“幻灯片播放”选项，就可以播放我们刚刚配置的幻灯片了。

6.3 配置主机资产信息

我们知道，Zabbix 系统具有主机资产管理的功能。当需要使用 Zabbix 系统对主机进行资产管理时，需要在配置主机信息时，配置相应主机的资产信息。在 Zabbix 系统中，主机的资产信息可以通过两种方式录入到系统中。第一种方法是，在主机上创建监控项目，通过监控项目，让 Zabbix 系统自动采集主机的资产信息。关于这种方法，在前面章节的介绍中，已经实际动手配置过这样一个监控项目。虽然让 Zabbix 系统自动采集主机的资产信息比较省事，但是，很显然 Zabbix 系统无法采集到主机资产的所有信息。例如，主机的编号、购买时间、资产标签、主机位置等，这些资产信息一般来说很难通过 Zabbix 系统自动采集。第二种方法是，在配置主机信息时，通过配置主机表单页面上的“主机资产”选项卡，手工录入主机的资产信息。通过这种方法配置主机资产信息比较灵活，可以完整地配置每台主机的资产信息。但是，这种方法也有一些缺点，那就是工作量比较大，不能实现自动化，很容易出错。

不管是通过自动方式录入主机资产信息，还是通过手工方式录入主机资产信息，对于一台主机，这两种方式只能二选一，不能既选择自动方式又选择手工方式录入主机资产信息。当使用自动方式录入主机资产信息时，Zabbix 系统的被监控设备代理为我们提供了一些用于获取主机硬件信息的监控项目关键字，详情请参阅本书的附录 C。

配置完主机的资产信息后，可以依次选择菜单项“资产记录”→“数据总览”或“资产记录”→“主机”查看系统中主机的资产信息。其效果如图 6-10 所示。

主机	组	名称	类型	操作系统	编号1	标签	网卡物理地址1
天津监控服务器	Linux服务器		服务器	CentOS 5.7 x86_64	tj-653	YL-I-S2013	3C:E5:A6:B6:B1:95
192.168.7.20	Linux服务器		服务器	CentOS 6.4 x86_64	nj-401	YL-I-S2008	D0:67:E5:EB:40:26
192.168.7.23	Linux服务器		服务器	CentOS 6.4 x86_64	Sh-nh-201	YL-I-S2006	F0:7B:CB:62:C0:F1

图 6-10 查看主机资产信息

6.4 配置认证方式和脚本

默认情况下，当用户登录时，Zabbix 系统是使用其自身数据库中所记录的用户账号信息对用户身份进行验证的。但是，Zabbix 系统还可以支持使用非系统自身数据库中的账号信息对用户的身份进行验证。本节将介绍如何修改 Zabbix 系统的认证方式。同时，还将介绍如何在 Zabbix

系统中配置和使用脚本。

6.4.1 配置认证方式

所谓认证方式是指用户在登录时，系统采用哪种方式来验证用户身份的合法性。在 Zabbix 系统中，除了可以使用系统默认的内部方法验证用户身份，还支持通过 LDAP 协议和 HTTP 两种方式验证用户身份。要修改系统的用户身份验证方式，依次选择菜单项“高级配置”→“认证方式”，系统将打开“配置认证方式”页面，如图 6-11 所示。

默认认证方式

内部

LDAP

HTTP

LDAP主机

192.168.5.139

端口

389

Base DN

dc=zabbix,dc=com

Search attribute

uid

Bind DN

cn=root,dc=zabbix,dc=com

Bind password

验证测试

[must be a valid LDAP user]

登录

Admin

用户密码

图 6-11 配置认证方式表单页面截图

如果要将 Zabbix 系统的认证方式改为“内部”或“HTTP”方式，则只需在如图 6-11 所示的“配置认证方式”页面上，单击“内部”或“HTTP”按钮，并单击页面下部的“保存”按钮即可。但是，不管使用哪种认证方式，在 Zabbix 系统中还是需要创建相应的用户的。这里所说的修改系统的认证方式，是指用户在登录到 Zabbix 系统时，系统通过哪种方式来确认用户所输入的账号和密码是正确的。但是，对于用户所具有的权限和所属于的用户组等，还是需要通过 Zabbix 系统内部账号来管理的。

当将认证方式调整为 HTTP 时，只需在调整之前配置好 Web 服务器的认证功能，并且针对 HTTP 认证创建好相应的用户账号信息即可。相应的，如果将认证方式修改为 LDAP 协议方式，则需要提前配置好相应的 LDAP 服务器，并且需要保证 Zabbix 系统能够正常访问该台服务器上的 LDAP 服务。

如果打算将认证方式修改为 LDAP 方式，则需要填写如图 6-11 所示的表单。该表单上各表单项的作用和含义如表 6-4 所示。

表 6-4 配置LDAP认证方式表单项列表

表单项	描述
LDAP主机 (LDAP host)	用于指定Zabbix系统将要访问的LDAP服务器的主机名或IP地址。该服务器上应存有用于登录认证的用户账号和密码信息，并且Zabbix系统要能通过LDAP协议获取到这些信息
端口 (Port)	用于指定LDAP服务的端口号，默认值是389
基础区别名称 (Base DN)	用于指定系统从哪个节点开始搜索所需要的账号信息。一般形式为如图6-11所示的“dc=zabbix,dc=com”形式
搜索的属性 (Search attribute)	用于指定系统在哪个属性上查找账号信息。对于Linux系统下的Open LDAP，这个表单项里的内容一般填写uid；对于使用Windows操作系统下的Active Directory，则这个表单项里的内容一般填写sAMAccountName

续表

表单项	描述
绑定的区别名称 (Bind DN)	用于指定Zabbix系统使用什么区别名称去连接LDAP服务
绑定的密码 (Bind password)	用于指定Zabbix系统使用上述的区别名称连接LDAP服务时，所使用的用户密码
登录 (Login)	这个表单项是用于输入测试信息的，用来测试Zabbix系统通过LDAP认证方式进行用户身份认证是否正常。所以，这个表单项里需要输入的内容就是用户可以登录到Zabbix系统的用户账号。当测试不通过时，系统将无法将认证方式修改为LDAP方式
用户密码 (User Password)	用于指定上述测试用户账号对应的密码。该密码即为LDAP服务所提供的userPassword属性中所记录的内容

6.4.2 配置脚本

通过前面的介绍我们知道了，可以通过外部脚本采集监控项目的数据；也可以在动作中配置脚本并将它作为远程命令在远程主机上执行；还可以将脚本配置成一个消息介质，让它在必要的时候给我们发送消息。所有这些脚本都是在系统中配置好后，由 Zabbix 系统在满足一定的条件时自动调用和执行。那么，能不能在 Zabbix 系统中定义和配置一些脚本，不由 Zabbix 系统自动调用和执行，而是在需要的时候手工执行呢？答案是肯定的，本节将介绍如何配置和使用这样的脚本。

依次选择菜单项“高级配置”→“脚本”，就可以看到如图 6-12 所示的系统中已存在的脚本信息列表。

名称	类型	执行目标	命令	用户组	主机组	主机访问
Detect operating system	脚本	服务器	sudo /usr/bin/ncmp -G (HOST.CONN) 2>&1	Zabbix administrators	全部	读
Ping	脚本	服务器	/bin/ping -c 3 (HOST.CONN) 2>&1	全部	全部	读
Traceroute	脚本	服务器	/usr/bin/traceroute (HOST.CONN) 2>&1	全部	全部	读

图 6-12 脚本列表

当在列表页面上单击“新建脚本”按钮，或单击列表上某个脚本名称上的超链接时，系统将会打开如图 6-13 所示的配置脚本表单页面。

名称

测试外网连接是否正常

类型

脚本

执行目标

☐ Zabbix agent

☒ Zabbix 服务器

命令

/bin/ping -c 4 www.baidu.com

描述

用户组

全部

主机组

Zabbix servers

需要主机权限

读

应用确认

☒

确认文本

执行脚本会影响监控系统性能，确认要执行么？

确认测试

图 6-13 配置脚本表单页面截图

图 6-13 所示的表单页面上部分表单项的作用和含义如表 6-5 所示。

表 6-5 配置脚本表单项列表

表单项	描述
名称 (Name)	用于指定脚本的名称，脚本名称在系统中不得有重复
类型 (Type)	用于指定脚本的类型，可选择类型有“脚本”和“IPMI”
执行目标 (Execute on)	用于指定脚本是通过Zabbix服务器执行，还是由运行在被监控主机上的被监控设备代理进程执行。很显然，如果选择通过Zabbix服务器执行，则在Zabbix服务器的指定路径下需要存在对应的脚本程序，且运行Zabbix服务器端进程的用户需要具有执行这个脚本的权限；同样，如果选择通过Zabbix Agent来执行指定的脚本，则在被监控主机上的指定路径下需要存在对应的脚本程序，且运行Zabbix Agent的用户（一般为Zabbix用户）需要具有执行这个脚本的权限。另外，如果选择通过Zabbix Agent执行指定的脚本程序，则被监控主机上被监控设备代理组件的配置文件zabbix_agentd.conf中的EnableRemoteCommands配置项要被设置为“1”，并需要重启zabbix_agentd进程
命令 (Commands)	用于配置需要被执行的具体命令。注意，配置命令时，需要填写完整的路径，否则Zabbix系统可能无法找到指定的命令。在命令行中可以支持使用以下这些宏变量：{HOST.CONN}、{HOST.IP}、{HOST.DNS}、{HOST.HOST}和{HOST.NAME}等。注意，如果宏变量所指代的值包含空格，则在命令行中使用该宏变量时，宏变量需要用引号引起来
用户组 (User groups)	用于指定属于哪些用户组的用户具有执行该脚本程序的权限
主机组 (Host groups)	用于指定在属于哪个主机组中的主机上可以执行该脚本程序
需要主机权限 (Required host permissions)	用于指定，如果某个用户需要在一台主机上执行该脚本，则它至少需要具有对这台主机的读取权限还是写入权限。例如，某个用户虽然具有执行某个脚本的权限，也就是它是属于上述“用户组”表单项中所指定的用户组中的用户，但是，它对某台主机不具有写权限，而如果要在这台主机上执行该脚本，则需要用户具对这台主机的写权限，此时，该用户就无权在对应的主机上执行该脚本
启用确认 (Enable confirmation)	如果选中这个复选框，则当用户在执行该脚本时，系统会弹出如图6-14所示的提示框，以让用户确认是否执行对应的脚本
确认文本 (Confirmation text)	用于配置，当系统弹出如图6-14所示的提示框时，系统显示在提示框中的内容

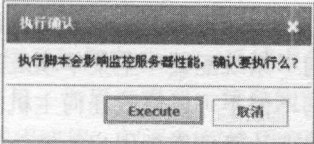


图 6-14 执行脚本提示框截图

6.5 配置用户及用户组

与其他绝大多数应用系统类似，在 Zabbix 系统中也有用户和用户组的概念。如前文所述，即使不使用 Zabbix 系统内部认证方式，而改用 HTTP 或 LDAP 认证方式，在 Zabbix 系统中还是需要创建相应的用户和用户组的。只是，用户在登录到 Zabbix 系统时，所使用的登录密码不

是 Zabbix 系统数据库中保存的密码而已。这是因为，在系统实施访问控制时，例如，哪些用户不可以修改或者查看哪些主机的配置信息等，Zabbix 系统需要使用到其内部数据库中所记录的用户账户信息。同时，当有动作被触发，需要给用户发送报警信息时，报警信息所发送的目标消息介质也是针对 Zabbix 系统内部用户账号配置的。

6.5.1 用户类型及用户权限

在 Zabbix 系统中，对用户权限是从两个方面进行管理和控制的。一方面，系统可以针对用户所能访问到的具体监控资源进行控制。例如，同是某种类型的用户，用户 A 可能具有修改主机 a 的配置的权限，而用户 B 则不具有相应的权限。另一方面，系统通过管理和控制用户所能访问的菜单项，以达到对用户的访问权限进行管理和控制的目的。在这个方面，系统主要是基于用户类型，来控制用户可以访问的菜单项。而在 Zabbix 系统中，任何用户必须是以下三种用户类型之一。因此，一个用户它既不可以同时属于两种或两种以上的用户类型，也不可以是下面所述的三种用户类型之外的其他用户类型。具体来说，Zabbix 系统中的三种用户类型及它们所具有的不同权限如下所述。

1. 普通类型用户 (Zabbix User)

属于普通用户类型的用户，具有访问 Web 组件上的“状态统计”、“资产记录”和“系统报告”等主菜单项及其子菜单项的权限。但是，默认情况下，这种类型的用户不具有访问系统中任何监控资源的权限。也就是说，在默认情况下，这种类型的用户既不可以查看系统中任何主机的监控数据、数据图、触发器状态等，更不具有修改系统中任何主机、监控项目、触发器和数据图配置的权限。如果给这类用户显式地赋予读取某些主机或主机组的权限，则相应的用户就可以查看对应主机的监控数据、数据图、触发器状态等监控资源信息。但是，即使将某些主机或主机组的写入权限赋给了普通类型的用户，该类型的用户也不具有修改对应主机及相关监控资源配置的权限。换句话说，普通类型的用户，无论额外给它赋予何种权限，它都不具有修改任何主机、数据图、触发器或其他监控资源配置的权限。

2. 管理类型用户 (Zabbix Admin)

这种类型的用户，除了具有访问普通类型用户所能访问到的“状态统计”、“资产记录”和“系统报告”等系统菜单项权限，还具有访问“系统配置”主菜单项及其下子菜单项的权限。但是，这种类型的用户不具有访问“高级配置”主菜单项及其下子菜单项的权限。

与普通类型用户类似，默认情况下，管理类型用户不具有访问系统中任何监控资源的权限。换句话说，虽然这类用户具有访问“系统配置”主菜单项及其下子菜单项的权限，但是，如果不给它赋予访问主机的权限，则它不但不可以修改任何主机的配置信息，而且也不可以查看任何主机上的监控数据。因此，如果要想让管理类型用户能够访问系统中某些监控资源，则就需要显式地给其赋予访问这些监控资源的权限，它才可以访问这些监控资源。

在 Zabbix 系统中，用户针对具体的监控资源具有“读”和“写”两种权限。而“读”和“写”权限的主要区别体现在：如果用户对某些监控资源只具有“读”权限，那么它就只能查看这些监控资源的监控数据、数据图、触发器状态等信息，却不可以修改主机、触发器或数据图的配置；而如果用户具有对某些监控资源的“写”权限，则它不但可以查看这些监控资源的监控数据、数据图、触发器状态等信息，而且还可以修改这些监控资源的配置。但前提条件是该用户需要具有访问“系统配置”菜单项的权限。

3. 超级管理员用户(Zabbix Super Admin)

在 Zabbix 系统中, 超级管理员用户是具有最高权限的用户。超级管理员用户不但具有访问任何系统菜单的权限, 而且也具有对系统中任何资源的“读”和“写”权限。不仅如此, 对于超级管理员用户, 我们不能回收它的某些权限, 它具有“天生”的最大权限。换句话说, 我们不能让某个超级管理员用户不可以访问某台主机, 或禁止它修改某台主机的配置。

与许多应用系统中规定超级管理员账号只能有一个不同, 在 Zabbix 系统中, 可以创建多个超级管理员账号, 而且它们所拥有的权限是一模一样的。Zabbix 系统中所采用的这种多超级管理员账号机制, 虽然在某些时候会给我们带来工作上的方便, 例如, 当有多个同事同时管理同一台 Zabbix 服务器时, 就可以分别为这些同事在系统中创建一个超级管理员账号, 然后每个同事分别用各自的账号登录到系统中进行系统管理; 但是, 使用这种多超级管理员机制, 如果权限管理不善的话, 很容易带来安全上的风险。

对于新安装部署的 Zabbix 系统, 系统默认会自动创建两个用户, 它们是 Admin 和 Guest 用户。其中, 用户 Admin 属于超级管理员用户, 它具有 Zabbix 系统中的最大权限。而 Guest 用户属于来宾用户, 默认情况下, 这是一个非常特殊的普通用户。说它特殊, 是因为来宾用户不需要登录就可以具有一般普通用户所具有的权限。当然, 前提条件是需要启用这个来宾用户账号。系统所创建的来宾用户账号在默认情况下属于普通用户, 但是, 可以将来宾用户账号设置成管理用户或超级管理员用户。当把来宾用户账号设置成管理用户或超级管理员用户时, 它就具有管理用户或超级管理员用户所具有的一切权限。因为来宾用户并不需要使用密码登录就可以直接操作 Zabbix 系统, 而且这个账户也不可以被设置密码, 所以除非有特殊需要, 建议一般情况下禁用该用户账号, 更不应该将来宾用户设置成管理用户或超级管理员用户, 以免给系统带来很大的安全风险。

前面介绍过, 在 Zabbix 系统中, 可以对用户针对具体的监控资源进行权限配置和管理。但是, 这种权限的配置和管理不能直接针对用户, 而只能针对用户组。也就是说, 不能直接将某个主机或主机组的“读”或“写”权限直接赋予给某个用户, 而只能将该权限赋予给某个用户组, 然后, 属于这个用户组的用户都具有相应的权限。因此, 如果要想将某个主机组的读或写权限赋予给某个用户, 则只需将该用户添加到具有相应权限的用户组中即可。类似的, 对于主机读、写权限的管理和分配, 也是在主机组层面上进行的, 即不能直接将某台主机的“读”或“写”权限赋予给某个用户组, 而只能将某个主机组的“读”或“写”权限赋予给某个用户组。

6.5.2 配置用户组

在前面章节中, 已经详细介绍了如何在 Zabbix 系统中创建用户账号, 这里不再复述如何在 Zabbix 系统中创建和配置用户账号。如果有需要, 可参阅前面章节中相关的内容。接下来介绍如何在 Zabbix 系统中创建和配置用户组, 以及如何给用户组赋予权限。

要创建和配置用户组, 依次选择菜单项“高级配置”→“用户”, 系统将打开显示当前系统中已存在的用户组信息的列表页面。如果要创建一个新的用户组, 则单击列表页面右上部的“新建用户组”按钮。而如果要修改一个已存在的用户组的配置信息, 则在列表页面上找到要修改的用户组, 并单击用户组名称上的超链接。不管是新建一个用户组, 还是编辑一个已存在用户组的配置信息, 系统都将打开如图 6-15 所示的“配置用户组”表单页面。

从图 6-15 中可以看出, “配置用户组”表单页面上有两个选项卡, 分别是“用户组”和“权限”选项卡。其中, “用户组”选项卡用于配置用户组的通用信息, 例如用户组的名称、隶属

于对应用户组的用户等。而“权限”选项卡则用于配置属于对应用户组的用户所具有的权限。

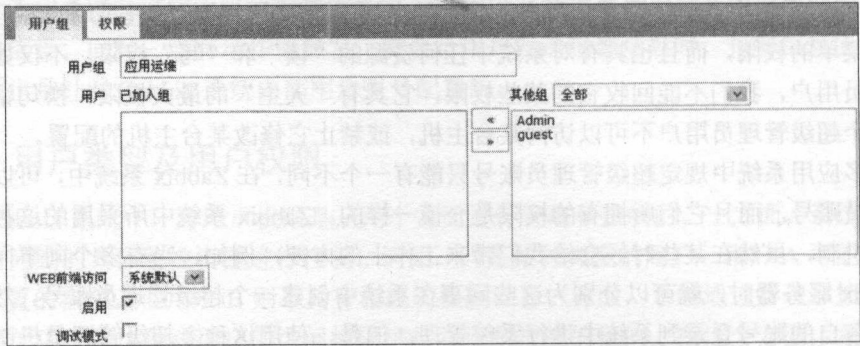



图 6-15 配置用户组表单页面截图

“用户组”选项卡上的表单内容比较简单，其中：

- ❑ “用户组 (Group name) ” 表单项。用于配置用户组的名称。用户组的名称在系统中不得有重复。
- ❑ “用户 (Users) ” 表单项。用于配置用户组中所包含的用户。从右边多选框中选择需要添加到当前用户组的用户，然后单击  按钮，就可以将所选定的用户添加到当前用户组中。
- ❑ “调试模式” 复选框。若勾选上这个复选框，则属于该用户组中的所有用户都开启调试模式。当用户开启了调试模式，则当用户登录到 Web 前端页面后，在系统的每个页面的右上角都会显示一个“调试”超链接。单击这个超链接，系统就会显示 Web 前端页面的调试信息。调试信息包括页面执行的总时间、SQL 查询语句总的执行时间、系统代码的执行过程以及系统所执行的每条 SQL 语句等，如图 6-16 所示。

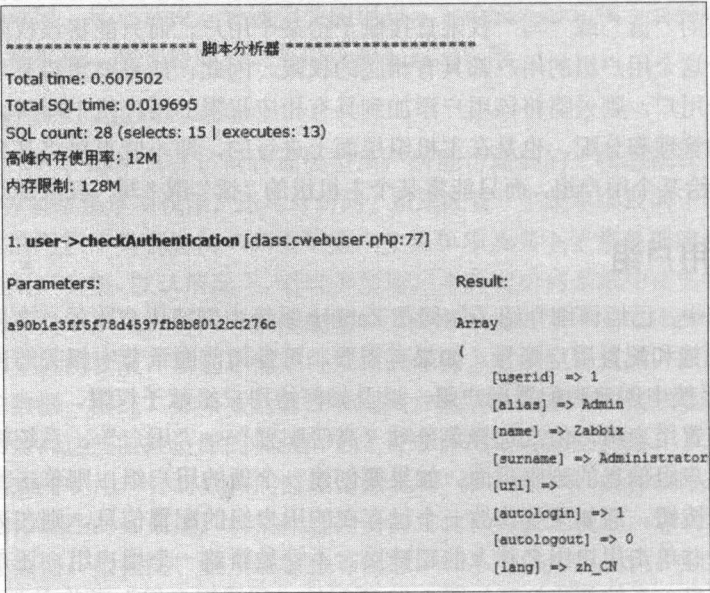


图 6-16 系统调试页面截图

单击“权限”选项卡，就可以为用户组配置主机组权限。这个选项卡有两大组多选框，分

别为“分配调整权限（Composing permissions）”和“最终获得权限（Calculated permission）”多选框。“分配调整权限”多选框组由“读写（Read-write）”、“只读（Read only）”和“禁止（Deny）”多选框组成。每个多选框下面都有“添加（Add）”和“删除选中项（Deleted selected）”两个按钮。单击“读写”多选框下面的“添加”按钮时，就可以给当前用户组添加新的具有读和写权限的主机组。而如果在“读写”多选框中选中一个或多个主机组，并单击该多选框下面的“删除选中项”按钮，就可以将选中主机组的读写权限从当前用户组中收回。相应的，当单击“只读”多选框下面的“添加”或“删除选中项”按钮时，则可以给当前用户组添加或删除指定主机组的只读权限。

而“最终获得权限”组中的各个多选框，则用于显示当前用户组最终获得读写、只读和禁止权限的主机和主机组列表。这些列表的内容是系统根据“分配调整权限”组中各个多选框中的内容自动计算出来的。因此，这些多选框中的内容会随着“分配调整权限”组中各多选框内容的变化而变化。

6.6 配置 IT 服务

与系统和网络维护人员主要关心诸如磁盘空间、系统负载情况、网络可用性等具体的监控细节不同，组织中某些高层领导，比如一个公司的总经理或者运维部门的运维总监等，他们可能更关心的是 IT 基础设施的“高级”监控信息，比如，IT 部门或服务商所提供的某个服务在某段时间内的可靠性、发现 IT 基础设施中的薄弱环节，以及监控 IT 服务的 SLA 信息等其他“高级”的 IT 监控信息。关于以上这些信息的监控，在 Zabbix 系统中都是通过 IT 服务功能模块完成的。

整个 IT 服务是一个由不同服务组成的层次结构树，它的根是 root，其他服务都是它的子节点。该结构树中的每个节点都有自己的状态属性，状态是通过一定的算法计算出来的，并且按照一定的算法传导到它的上级节点上。最低层的服务是系统中配置的触发器，它的状态由对应的触发器状态所决定。需要注意的是，未定义级别的触发器或信息级别的触发器，它们的状态不会影响 SLA 的计算结果。

要配置 IT 服务，依次选择“系统配置”→“IT 服务”菜单项，系统将打开“配置 IT 服务”页面，如图 6-17 所示。

配置 IT 服务		
IT 服务		
服务	状态统计	触发器
root		
[-] 活动网站系统	Problem, if at least one child has a problem	-
[-] 数据库系统	Problem, if at least one child has a problem	mysql_3306端口连接3次不通
[-] 应用系统	Problem, if at least one child has a problem	ssh_2208端口连接5分钟不通
[-] 应用系统	Problem, if at least one child has a problem	mysql_3306端口连接3次不通
[-] 应用系统	Problem, if at least one child has a problem	zabbix_10051端口连接3次不通

图 6-17 配置 IT 服务表单页面截图

在图 6-17 中，当在某个级别的服务名称上单击鼠标右键时，系统将会弹出一个菜单。通过

这个菜单，可以在所选定的级别上创建一个新的子节点，或者编辑所选定的服务。如果所选定的服务没有任何下一级子节点，那么，也可以通过这个弹出式菜单删除所选定的服务。也就是说，如果某个服务还有下级子节点，则不可以将它删除。如果要删除某个服务，需要首先将该服务下的所有子节点都删除了，才能删除该服务。不管是创建一个新子节点，还是编辑一个已存在的服务，系统都会为我们打开如图 6-18 所示的“配置 IT 服务”表单页面。

The screenshot shows the 'Configure IT Service' form with the following fields and values:

- 名称 (Name):** Web服务器
- 上一级服务 (Parent service):** 活动网站系统
- Status calculation algorithm:** Problem, if at least one child has a problem
- Calculate SLA, acceptable SLA (in %):** 99.050
- 触发器 (Trigger):** 192.168.10.23:到南京机房网络不通
- Sort order (0->999):** 0

图 6-18 配置 IT 服务表单页面截图

由图 6-18 中可以看出，“配置 IT 服务”页面由三个选项卡组成，它们是“服务”、“依赖”和“时间”选项卡。其中，“服务”选项卡上的表单用于配置 IT 服务的通用信息，如服务的名称、状态算法、可接受的 SLA 以及反映服务状态的触发器等，该选项卡上部分表单项的含义和作用如下所述：

- ❑ **状态计算算法 (Status calculation algorithm) 表单项。**用于选择计算当前服务状态的算法。这个表单项有三个备选项，它们是：“不计算 (Do not calculate)”，表示不通过计算该服务的子节点的状态来获得该服务的状态。一般这种情况下，该服务的状态完全由与它所关联的触发器的状态所决定。“只要有一个子节点的状态是问题状态，则该服务的状态便计为‘问题’状态 (Problem, if at least one china has a problem)”，如果选择这个选项，则该服务的子节点服务和其所依赖的服务中只要有一个服务处于问题状态，该服务便计为问题状态。“所有子节点的状态为问题时才将该服务计为问题状态 (Problem, if all children have problems)”，如果选择这个选项，则只有当该服务下的所有子节点服务和其所依赖的服务都处于问题状态时，该服务才会被计为问题状态。
- ❑ **可接受的 SLA[Calculate SLA,acceptable SLA(in %)]表单项。**用于配置该服务在服务等级协议中所规定的可以接受的服务可用率。这个数值一般是与服务提供商共同商定下来的。当通过“状态统计”→“IT 服务”菜单项查看服务的可用率时，如果对应服务实际计算和统计出来的服务可用率低于这个表单项所配置的数值时，则系统将会以红色字体显示对应服务的实际可用率。
- ❑ **触发器 (Trigger) 表单项。**用于选择该服务所关联的触发器。当然，如果所配置的服务不是处于 IT 服务结构树中最底一层的服务，则该服务也可以不关联触发器，此时，该服务的状态是由其下的子节点的服务的状态根据状态计算算法计算出来的；否则，该服务的状态就由与它所关联的触发器的状态所决定。当所配置的服务是处于 IT 服务结构树中最底一层的服务时，则该服务必须要与一个触发器相关联；否则，对应服务的状态就没有任何实体来反映它。

❑ **排序 (Sort order) 表单项。**用于指定所配置的服务处在 IT 服务树中的位置，数值越小，则排列的位置越靠前。

很显然，“依赖”选项卡上的表单是用于配置当前服务依赖关系的，当选中“依赖”选项卡时，系统将显示如图 6-19 所示的表单内容。

服务	依赖	时间												
	<div>依赖于</div> <table><thead><tr><th>服务</th><th>软件</th><th>触发器</th><th>动作</th></tr></thead><tbody><tr><td>MySQL 服务器</td><td><input type="checkbox"/></td><td>MySQL 服务端口不正常</td><td>移除</td></tr><tr><td colspan="4">添加</td></tr></tbody></table>	服务	软件	触发器	动作	MySQL 服务器	<input type="checkbox"/>	MySQL 服务端口不正常	移除	添加				
服务	软件	触发器	动作											
MySQL 服务器	<input type="checkbox"/>	MySQL 服务端口不正常	移除											
添加														

图 6-19 配置服务依赖关系表单页面截图

单击“依赖”选项卡上的“添加”超链接，可以给当前服务配置所依赖的服务。在实际工作中，我们经常会遇到一个服务的可用性状态会依赖于其他一个或多个服务的可用性状态的情况。例如，一个网站的可用性状态会依赖于网络连通性状态、Web 服务器的可用性状态、数据库的可用性状态、中间件的可用性状态等。这个时候，为了真实地反映一个网站服务的可用性，就需要将网络服务、Web 服务、数据库服务等服务添加为网站服务所依赖的服务。当一个服务的可用性状态依赖于其他一个或多个服务的可用性状态时，我们应优先将这些被依赖的服务添加为该服务的子节点服务。但是，在很多时候，一个服务往往会被多个服务所依赖。例如，前面所提到的网络连通性服务，它不但被网站服务所依赖，还会被邮件服务、FTP 服务等众多的服务所依赖。这个时候，如果按照我们前文所述的，应优先将被依赖的服务添加为依赖服务的子节点服务，那么，网络连通性服务就将会被添加为网站服务、邮件服务和 FTP 服务等众多服务的子节点服务。而这在 Zabbix 系统中是不被允许的，也就是说，在 Zabbix 系统中，一个服务不可以同时是多个服务的子节点服务。为了解决这个问题，Zabbix 系统中引入了软依赖的概念。你或许已经注意到了，在图 6-19 中有一个“软件”（这个表单项名称被汉化得非常不准确，笔者认为它应该叫做“软依赖”更合适一些）复选框，若勾选了这个复选框，则被依赖的服务与当前所配置的服务之间的关系是软依赖关系。当一个服务被软依赖于另一个服务时，则这个服务在服务结构树中的位置不会被改变，而只是建立起了与依赖于它的服务之间的依赖关系。而当不勾选“软件”这个复选框时，则依赖服务与被依赖服务之间所建立起来的是硬依赖关系。当一个服务与另一个服务之间是硬依赖关系时，则实际上对于系统来说，是将它们视为父节点（依赖于其他服务的服务为父节点）和子节点（被依赖的服务为子节点）之间的关系，并且被依赖的服务在服务结构树中的位置可能会被改变。

当一个服务软依赖于另一个服务时，虽然被依赖的服务在服务结构树中的位置不会被改变，但是，系统仍然会以它的名称，在依赖于它的服务的下面创建一个虚拟的子节点，该子节点的名称，系统将会以灰色字体来显示。而且，只拥有虚拟子节点的父节点是可以被删除的。当删除该类父节点时，它下面的虚拟子节点所对应的服务不会被一同删除。

“配置服务”表单页面上的第三个选项卡是“时间”选项卡。当选择“时间”选项卡时，系统将显示如图 6-20 所示的配置服务时间表单。

“时间”选项卡上的表单用于配置，服务在哪些时间段应该处于在线（也就是可用）状态。该表单上“新建服务时间”表单域中的单选列表具有三个选项，它们是：上线时间（Uptime），用于指定服务在线的时间范围；下线时间（Downtime），用于指定服务下线的范围，在这个时间范围内，服务的状态不影响 SLA 可用率的计算；最后一个选项是“单次下线时间（One-time

downtime)”，前面两个选项用于指定以一个星期为周期，服务上线和下线的时间范围，而这个选项则用于指定，单次服务下线的时间范围。与选择“下线时间”这个选项类似，如果选择“单次下线时间”这个选项，则在对应时间范围内，服务的状态不会影响 SLA 可用率的计算。

服务

依赖

时间

服务时间

类型

间隔

Note

动作

上线时间

周一 00:00 - 周五 23:59

移除

新建服务时间

上线时间

起始

周日

时间

截止

周日

时间

添加

图 6-20 配置服务时间表页面截图

完成 IT 服务的配置后，依次选择菜单项“状态统计”→“IT 服务”，就可以查看系统所统计和计算的整个 IT 服务的 SLA 可用率情况，如图 6-21 所示。

服务	状态	Reason	Problem time	SLA / 可接受的服务等级协议
root				
[-] 活动监控系统	OK	-		0.0000 100.0000/ 50.0000
[-] 数据库服务 - mysql 3306端口连续3次不通	OK	-		0.0000 100.0000/ 50.0000
[-] Web服务器	OK	-		0.0000 100.0000/ 99.0500
[-] zabbix服务数据库 - zabbix 10051端口连续3次不通	OK	-		0.0000 100.0000/ 99.0500
[+] OA系统	OK	-	1.8041	98.1959 / 99.0500
[-] 监控系统	OK	-		
[-] 数据库服务 - mysql 3306端口连续3次不通	OK	-		0.0000 100.0000/ 99.0500
[-] zabbix服务数据库 - zabbix 10051端口连续3次不通	OK	-		0.0000 100.0000/ 99.0500

图 6-21 IT 服务统计信息页面截图

注意：服务上下线时间的配置只对本服务有效，父节点所对应的服务不会因为子节点所对应的服务处在计划下线时间内，而不计算因为它处于问题状态下而带来的对 SLA 可用率的影响。因此，一般来说，如果在子节点上创建了计划上下线时间范围，那么在父节点上也应创建相应的上下线时间范围。

6.7 “常规”配置

“高级配置”→“常规”页面是用于配置与 Web 前端相关的各种设置的。在前面章节中，我们已经陆续对该页面上部分项目的配置作过一些介绍，下面我们将介绍该页面上其他项目的配置。

6.7.1 “图形界面（GUI）”配置

依次选择菜单项“高级配置”→“常规”，系统将打开“图形界面”页面，如图 6-22 所示。在“配置图形界面”页面标题栏的右边有一个下拉菜单，通过这个下拉菜单，可以选择所需要配置的项目。

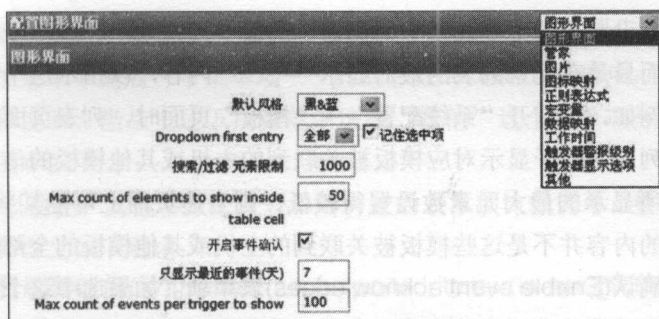


图 6-22 配置图形界面表单页面截图

“配置图形界面”页面上的表单内容比较简单，下面我们对这个表单上部分表单项的作用和含义做一个简单的说明。

- **默认风格(Default theme)表单项。**“默认风格”表单项用于指定用户在没有修改页面显示风格的情况下，系统将以何种风格显示 Web 前端页面。Zabbix 系统为我们预定义了“经典(Classic)”、“默认蓝色(Original blue)”、“黑和蓝(Black & Blue)”和“深橙色(Dark orange)”等几种 Web 前端页面显示风格供我们选择。当然，用户可以通过单击页面右上部的“配置(Profile)”超链接，自行修改 Web 前端页面的显示风格。
- **下拉菜单首个菜单项(Dropdown first entry)表单项。**这个表单项用于指定系统在显示下拉菜单时，下拉菜单的第一个选项是“全部(All)”还是“没有选中项(None)”菜单项。当选中“记住选中项(remember selected)”复选框时，则系统在显示下拉菜单时，会记住用户上次所选中的菜单项，并在用户下次再次打开对应页面时，默认会选中用户上次所选中的菜单项。否则，系统在显示下拉菜单时，不会记住用户上次所选中的菜单项，而是默认将“全部”或“没有选中项”菜单项设置为被选中的菜单项。
- **搜索/过滤元素限制(Search/Filter elements limit)表单项。**这个表单项用于指定列表类页面，例如“状态统计”→“事件”页面、“系统配置”→“主机”等页面所显示的最大记录条数。如果系统从数据库中所能查询到的记录数大于这个表单所设置的最大显示记录条数，则 Zabbix 系统将最多只显示这个表单项所配置的记录条数，超过的部分将不在页面上显示出来。但是，系统会在过滤器上部的位置，显示如“正在显示 1 to N of N+已经到 (Displaying 1 to N of N+ found)”的提示信息。“N”即为这里通过“搜索/过滤元素限制”表单项所配置的数字。如果使用了过滤器进行过滤后所得到的数据条数仍然大于这个表单项所设置的条数时，则系统只显示这个表单项所配置的前 N 条记录。例如，如果所设置的搜索/过滤元素限制数为 50，而实际数据经由过滤器过滤后，所得到的数据的条数仍然大于 50，则系统只显示前 50 条数据。而且，在 Zabbix 系统中，列表类的页面是不支持翻页功能的。所以，这一点需要特别注意，如果在系统中所配置的监控元素比较多，则需要相应的将这个表单项的数值设置得稍微大一些，以免因为数据没有在页面上显示出来，而误认为系统中没有配置相应的监控元素。
- **每个单元格所显示的最大元素数(Max count of elements to show inside table cell)表**

单项。用于设置在每个单元格中所能显示的最多元素数，超过这个数字的元素将不会被显示，而只是在元素列表的最后显示“……”内容，以指示这个单元格里还有更多的内容。例如，当打开“系统配置”→“模板”页面时，列表页面上有一个叫作“关联到”的列，它用于显示对应模板被关联到的主机或其他模板的信息列表。如果将每个单元格所显示的最大元素数设置得较低，则上述页面上可能某些模板的“关联到”列所显示的内容并不是这些模板被关联到的主机或其他模板的全部数据。

- ❑ **开启事件确认(Enable event acknowledges)表单项。**如果选中这个复选框，则表示开启事件确认功能。当开启了事件确认功能，则在“状态统计”→“事件”页面上会显示事件的确认状态。此时，如果事件未被确认，单击确认列里的超链接，则可以对相应的事件进行确认。而如果不开启事件的确认功能，则在“状态统计”→“事件”页面上不会显示“确认”超链接，故不可以对事件进行确认操作。
- ❑ **只显示最近的事件 (天) [Show events not older than (in days)]表单项。**用于指定在“状态统计”→“触发器”页面上显示最多多少天的事件信息。
- ❑ **每个触发器最多显示多少条事件信息(Max count of events per trigger to show)表单项。**用于指定在“状态统计”→“触发器”页面上，每个触发器上显示多少条事件信息。

6.7.2 “管家 (Housekeeper)” 配置

当在“配置图形界面”页面标题栏的右边下拉菜单里选择“管家 (Housekeeper)”菜单项时，系统将打开“配置管家”页面，如图 6-23 所示。“管家”是一个被 Zabbix 服务器端组件定时执行的进程，该进程用于彻底删除过期的和被用户删除的信息（例如，因低级自动发现规则的改变，而需要由系统自动删除的监控项目等）。

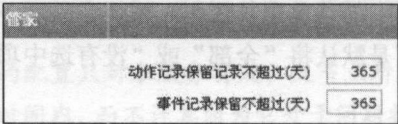


图 6-23 配置管家表单页面截图

如图 6-23 所示，配置管家表单页面上的内容非常简单，这个表单总共只有两个表单项。它们的作用和含义也十分简单，其中，“动作记录保留不超过 (天) [Do not keep actions older than (in days)]”表单项，用于指定在系统中保留最长多少天的动作被触发的历史记录，超过这个时间的动作被触发的历史记录，将会被“管家”进程自动给删除掉。而“事件记录保留不超过(天)[Do not keep events older than (in days)]”表单项，则用于配置在系统中保留最长多少天的事件记录，过期的事件记录将被“管家”进程自动给删除掉。

6.7.3 “其他参数 (Other)” 配置

通过“配置图形界面”页面标题栏右边的下拉菜单，还可以选择“图片 (Images)”、“图标映射 (Icom map)”、“正则表达式 (Regular expressions)”、“宏变量 (Macros)”、“数据映射 (Value mapping)”、“工作时间 (Working time)”、“触发器级别 (Trigger severities)”和“触发器显示选项 (Trigger displaying options)”等菜单项，以分别对这些项目进行配置。上述这些菜单项所对应项目的配置非常简单，而且有一些项目的配置在前面章节中做过一些介绍，

故在此不再一一说明它们的配置方法。

这里对上述下拉菜单中最后一个菜单项——其他参数（Other）菜单项，所对应项目的配置作一个简单的介绍和说明。当从上述下拉菜单中选择“其他”这个菜单项时，系统将打开“其他参数”页面，如图 6-24 所示。

从图 6-24 中可以看出，其他参数配置表单所对应的表单项比较简单，整个表单只有 4 个表单项，它们的作用和含义如下所述：

- ❑ **不支持项目刷新时间(秒)[Refresh unsupported items (in sec)]表单项。**当某些监控项目因为某种原因，例如参数配置错误，或者被监控对象从被监控主机上被移去而导致不被支持时，不被支持的监控项目，在主机监控项目列表页面的“状态”列里所显示的状态为“不支持”状态。Zabbix 系统会定期检查这些不被支持的监控项目，并且尝试将它们的状态转变为被支持状态。“不支持项目刷新时间”表单项就是用于配置系统每隔多长时间（单位是秒）检查一次这些不被支持的监控项目，并且尝试将它们的状态转变为被支持状态的。当这个表单项被填写为 0 时，则表示禁用 Zabbix 系统自动检查并尝试转变不被支持监控项目状态的功能。此时，系统中如果有不被支持的监控项目，就需要我们手动去尝试启用它们了。但是，如果不被支持的监控项目是通过 Zabbix 服务器代理完成监控数据采集的，则这个配置对于这些不被支持的监控项目不起作用。此时，Zabbix 服务器代理会每隔 10 分钟检查一次不被支持的监控项目，并尝试将它们的状态由不支持状态转变为支持状态，而且这个检查的时间间隔对于 Zabbix 服务器代理来说是不可配置的。



图 6-24 其他参数配置表单页面截图

- ❑ **自动发现主机组(Group for discovered hosts)表单项。**用于配置，对系统通过网络自动发现功能或被监控设备代理自动注册功能自动发现的主机，系统默认情况下将它们添加到哪个主机组。当然，我们在前面章节中介绍过，我们完全可以通过动作来实现将系统所发现的主机自动添加到指定主机组。
- ❑ **数据库下线通知用户组(User group for database down message)表单项。**用于指定当 Zabbix 服务器进程或其代理进程所使用的数据库服务，因为某种原因出现连接不上的故障时，系统给哪个用户组的用户发送通知信息。被指定用户组的所有用户，在 Zabbix 系统所使用的数据库服务出现故障时都会收到相关的报警信息，而且系统会使用用户所有可用的消息介质给用户发送报警信息，而不管用户所配置的消息介质可接受的报警级别以及该消息介质的激活时间。
- ❑ **日志记录不匹配的 SNMP traps 数据(Log unmatched SNMP traps)表单项。**用于配置当 Zabbix 系统接收到一个未知的 SNMP 陷入数据（未能匹配陷入的数据内容或相关主机上未配置相应的陷入类型监控项目等）时，是否在系统中记录相应的日志信息。

6.8 日常管理功能介绍

在前面章节中，对于独立服务器模式下 Zabbix 系统的 Web 前端页面上的各种配置做了比较详细的介绍和说明。本节将介绍 Zabbix 系统的 Web 前端组件为我们提供的日常管理和维护功能，以及它们的操作方法和操作过程。

6.8.1 批量更新 (Mass update)

在某些时候，我们可能需要对多台主机的某些属性进行修改。例如，由于业务的调整，要将某些主机从一个机房迁到另外一个机房，这个时候就需要对用于采集这些被迁移主机监控数据的 Zabbix 服务器代理节点进行修改和调整。此时，如果逐台修改这些被迁移主机所使用的 Zabbix 服务器代理节点，不但工作量巨大，操作起来重复单调，而且还很容易出错。而如果使用系统提供的“批量更新 (Mass update)”功能，则不但能够高效、快速地完成这部分工作，而且也不容易出错。

批量更新的操作方法比较简单，依次选择菜单项“系统配置”→“主机”；然后，在主机列表页面上选中需要批量更新属性的主机；最后从页面下部的下拉菜单中选择“批量更新 (Mass update)”菜单项，并单击“确认”按钮，系统将打开如图 6-25 所示的批量更新表单页面。

替换主机组		其他组	
<input checked="" type="checkbox"/>	已加入组	<input type="checkbox"/>	Discovered hosts
<input type="checkbox"/>	新建主机组	<input type="checkbox"/>	Linux servers
<input type="checkbox"/>	由代理节点监控	<input type="checkbox"/>	Linux服务器
<input type="checkbox"/>	状态	<input type="checkbox"/>	Templates
<input type="checkbox"/>	Link templates	<input type="checkbox"/>	Windows Servers
<input type="checkbox"/>	IPMI认证算法	<input type="checkbox"/>	Zabbix servers
<input type="checkbox"/>	IPMI特权级别		
<input type="checkbox"/>	IPMI用户名		
<input type="checkbox"/>	IPMI密码		
<input type="checkbox"/>	Inventory mode		

图 6-25 批量更新主机属性表单页面截图

从图 6-25 中可以看出，可以一次性对主机组、Zabbix 服务器代理节点、主机状态、所关联的模板等主机属性进行批量更新。当选中属性前面的复选框时，系统就会显示出修改对应属性所需要的表单项。例如图 6-25 中，选中了“替换主机组”这个属性的复选框，则系统显示“已加入组”和“其他组”两个多选框。通过这两个多选框，就可以对被选中主机所属于的主机组进行重新配置。

在批量更新主机属性页面上，可以一次更新主机的多个属性。但是，需要注意的是，如果通过批量更新方法修改了主机的某些属性，则对应主机上被更新属性的配置将会被覆盖掉。例如，有主机 A、B 和 C，原来它们均同时属于主机组 a、b 和 c，如果通过批量更新方法更新了它们所属于的主机组为 d、e 和 f，则这些主机将会从主机组 a、b 和 c 中被删除，并被添加到主机组 d、e 和 f 中。

6.8.2 维护模式

在实际工作中，很多时候需要计划性地去维护一些主机。很显然，在对主机进行计划性维护时，我们并不需要接收到关于它们的报警信息。此时，如果在具体实施主机计划性维护之前，在 Zabbix 系统中将所需要维护的主机设置成维护模式，就可以做到在主机维护期间系统不再发送报警信息了。但是，如果要让 Zabbix 系统在主机维护期间不发送相关的报警信息，仅将主机设置成维护模式还是不够的。读者或许还记得我们在介绍动作配置的时候介绍过，当在系统中创建一个动作时，系统会在我们所新创建的动作上自动地添加上一个触发条件“维护模式状态 not in ‘维护’”。只要动作中所配置的这个触发条件，没有被我们人为地删除掉，处于维护模式下的主机就不会引起系统中相关动作被触发，当然也就不会给用户发送报警信息了。

Zabbix 系统中主机的维护模式有两种类型，分别是带数据采集（With data collection）类型维护模式和不带数据采集（with no data collection）类型维护模式。所谓带数据采集类型维护模式是指，在该模式下，Zabbix 系统仍然会对处于维护状态下主机上的监控项目进行监控数据的采集；而相应的，不带数据采集类型维护模式是指，在该模式下，Zabbix 系统不会对处于维护状态下主机上的监控项目进行监控数据的采集。但是，所谓带数据采集或不带数据采集是针对 Zabbix 服务器而言的，如果被监控主机是通过 Zabbix 服务器代理组件采集监控数据的，则 Zabbix 服务器代理组件对处于维护模式状态下的主机仍然会正常地采集监控数据，并且会正常地将采集到的监控数据发送给 Zabbix 服务器进行处理。只是，如果被监控主机被配置成不带数据采集的维护模式，则 Zabbix 服务器会将接收到的相应数据丢弃不处理。

当有主机处于维护模式状态下时，Zabbix 系统的服务器端进程必须要处于运行状态下。Zabbix 系统服务器端组件中的定时器进程，会负责在每分钟的第 0 秒检查系统中维护模式的配置情况，并将维护模式处于激活状态下的主机设置成维护状态，或将结束维护模式状态的主机设置成正常状态。

当主机处在维护模式期间出现了故障，从而引起某些触发器被触发，且对应主机维护模式结束该故障还没有被恢复时，系统在对应主机维护模式结束后会正常地发送报警信息。虽然某些主机所包含的触发器表达式中可能调用了 `nodata()` 函数，但它不会因为这些主机被设置成了不带数据采集类型的维护模式，而使这些主机在维护状态期间不采集监控数据。而在维护模式结束后，相关的触发器会被触发，进而引起系统发送不必要的报警信息。调用了 `nodata()` 函数的触发器，只会在调用它的主机结束了维护状态之后，Zabbix 服务器针对该触发器所引用的监控项目都做过一次监控数据采集，且没有采集到相应的监控数据时才会被触发。

主机的维护模式周期配置比较简单，依次选择菜单项“系统配置”→“维护模式”，系统将打开显示当前系统中已存在的维护模式周期列表页面。如果需要对已存在的维护模式周期进行修改和编辑，则只需在上述页面上找到需要修改的维护模式周期，并单击其名称上的超链接。而如果需要新建一个维护模式周期，则只需单击上述页面上的“新建维护模式周期”按钮。不管是修改一个已存在的维护模式周期，还是新建一个维护模式周期，系统都将会打开如图 6-26 所示的“配置维护模式周期”表单页面。

由图 6-26 可以看出，在配置维护模式的表单页面上有三个选项卡，分别是“维护模式（Maintenance）”、“周期（Periods）”和“主机&主机组（Hosts & Groups）”。在这三个选项卡中，“主机&主机组”选项卡上表单项的内容最容易理解，它是用于配置在指定的时间段内哪些主机和属于哪些主机组中的主机被系统设置成维护模式。同时，该选项卡上的表单项操作也十分简单。因此，我们在此就不一一阐述它们的作用和含义了。

维护模式

周期

主机 & 主机组

名称

20140921全服更新

维护模式类型

采集数据

激活时间

21 / 09 / 2014 14 : 07

激活至

21 / 09 / 2014 20 : 07

描述

20140921全服更新

图 6-26 配置维护模式周期表单页面截图

“维护模式”选项卡上的“名称”、“维护模式类型”和“描述”等表单项的作用和含义比较容易理解。其中，“名称”即为维护模式的名称，可以是任意字符组成的字符串，但是要求在系统中不得有重复的维护模式名称。“维护模式类型（Maintenance type）”表单项，用于配置维护模式的类型，其类型应是上述两种类型（带数据采集类型和不带数据采集类型）之一。接下来，如果选中“周期”选项卡，则可以看到如图 6-27 所示的配置维护周期的表单页面。

周期

One time only

At 14:12 on 21 九月 2014

1h

编辑

每周

At 12:00 on 每 周一, 周三, 周五 of every week

1h

编辑

新建

删除选中项

编辑维护模式周期

周期类型

每周

周

1

Day of week

☒ 周一

☐ Tuesday

☒ 周三

☐ 周四

☒ 周五

☐ 周六

☐ 周日

At (hour:minute)

12 : 0

维护模式周期长度

0 天数 1 小时 0 分钟

图 6-27 配置维护周期表单页面截图

由图 6-27 可以看出，在一个维护模式中，可以配置多个维护周期。而且，还可以通过“周期类型（Maintenance type）”下拉列表，来指定周期的类型是单次的（One time only）、按每日的（Daily）、按每个星期的（Weekly）还是按每个月（Monthly）的。如果周期类型选择单次的，则表示系统会在周期所指定的时段内，将所指定的主机状态设置成维护状态，这种类型的周期比较好理解。而如果周期类型选择的是每个月，则系统提供两种方式来指定具体的日期。一种方式是指定具体的日期；另一方式是，通过指定月份中第几个星期的星期几的方式来指定具体的日期。而在说明每日和每周类型的周期之前，我们再回过头来看看图 6-26 所示的“维护模式”选项卡上的表单内容。

在“维护模式”选项卡上有两个表单项：“激活时间（Active since）”和“激活至（Active till）”。从这两个表单项的名称上可以很容易地猜出，它们是用于配置维护模式激活的时间范围的。这样，或许读者就会有疑问了，既然我们在前面介绍过，在一个维护模式中配置多

个周期，并且在所配置的周期时间范围内，系统会自动将指定的主机的状态设置成维护状态，那么，这里的激活时间范围起什么作用呢？前面介绍过，维护的周期可以按次（单次的）、日、星期和月来配置。这样，除了单次的周期不存在重复执行的情况，按日、星期或月配置的周期都会存在重复执行的情况。例如，在每天的 9:00~18:00 将某些主机的状态设置成维护状态等。但是，在很多时候，这类重复执行的周期也需要有一个时间范围，比如在 4 月 1 日到 4 月 7 日这段日期范围内，每天的 9:00~18:00 将某些主机的状态设置成维护状态等。而表单项“激活时间（Active since）”和“激活至（Active till）”即是用于指定这个时间范围的。在 Zabbix 系统中，也将这个时间范围称之为维护模式被激活的时间范围。“维护模式”选项卡上最后一个表单项“描述”，很显然是用于描述维护模式的。

介绍完“维护模式”选项卡上各表单项的内容，我们再回过头来看看前面尚未介绍完的另外两种周期类型——每日和每周。其实不管是每日还是每周类型的周期，其配置本身都不是很复杂，我们之所以颠过来倒过去地叙述，是因为在配置这两种类型的周期时都有一个很“奇怪”的表单项。我们再回过头来仔细看一下图 6-27，你会发现在图 6-27 中有一个表单项叫做“周”。类似的，当在周期类型表单项里选择“每日”时，会有一个叫做“天”的表单项。为什么有叫这么奇怪名称的表单项呢？它们又是什么含义以及有什么作用呢？如果简单地理解这里的“周”或“天”表单项的含义，可以将其理解为是间隔。即每隔多少天或多少周重复执行周期计划。但是，这里所配置的间隔又与前面所配置的激活时间的起始时间有关。如果激活时间的起始时间，小于周期中所定义的维护的开始时间，那么维护周期首次被执行的日期就是激活时间所指定的当天。当然，前提条件是，Zabbix 服务器的系统时间没有超过维护周期中所指定的结束时间。如果 Zabbix 服务器的系统时间超过了维护周期中所指定的结束时间，则维护周期的首次被执行，是在指定间隔的天数或周数之后。而如果激活时间的起始时间大于周期中所定义的维护的开始时间，则周期的首次被执行，是在“激活时间”中所指定日期之后的指定间隔天数或周数之后（也就是“天”或“周”表单项中所指定的天数或周数）。

下面举个具体的例子来说明上述的“天”和“周”表单项的作用。假如在“维护模式”选项卡上，将维护模式激活的起始时间设置为 2014 年 4 月 24 日的中午 12 时整。而在“周期”选项卡上创建一个“每日”类型的周期，并且将维护周期的开始时间设置为 23:00，同时配置周期长度为 1 小时，“天”表单项的内容填写为 2。则这个周期首次被执行是在 2014 年 4 月 24 日晚上 23 时，也就是在 2014 年 4 月 24 日晚上 23 时开始将指定的主机设置成维护状态，而到 2014 年 4 月 25 日零时将相应主机的状态恢复为正常状态。当然，如前文所述，如果要想 Zabbix 系统在 2014 年 4 月 24 日晚上 23 时的时候，将指定主机都设置成维护状态，则必须要满足，在配置该维护模式时，Zabbix 服务器的系统时间不能晚于 2014 年 4 月 25 日零时（也就是维护周期的结束时间）。而因为执行间隔被设置成 2 天（也就是“天”表单项里所填写的内容），所以，这个维护周期第二次被执行是在 2014 年 4 月 26 日的 23 时。但是，如果维护模式激活的起始时间不变，还是 2014 年 4 月 24 日中午 12 时，且周期长度仍为 1 小时，而将维护周期的开始时间修改为 10 时整，则这个周期首次被执行是在 2014 年 4 月 26 日上午的 10 时整，周期为一个小时，第二次被执行是在 2014 年 4 月 28 日的上午 10 时整。

配置维护模式的操作比较简单，这里不再详细介绍配置维护模式的具体步骤，读者只需结合上述介绍，实际动手配置一个维护模式，即可熟悉配置维护模式的具体操作过程和步骤。

注意：当一个触发器引用了多台主机上监控项目的数据时，只要这些主机中有一台主机不被设置成维护模式，则不管动作中有没有配置“维护模式状态 not in ‘维护’”触发条件，

相关动作都可能会被触发。换句话说，只要触发器表达式中引用了一台非维护模式状态下主机上的监控项目，则由该触发器的触发所产生的事件，进而引起动作的触发将不受“维护模式状态 not in ‘维护’”触发条件的影响。

6.8.3 事件确认

事件确认功能可以用于协同办公。例如，当有多位同事共同负责维护 Zabbix 系统时，一旦有多台被监控主机同时出现了问题，则可能有多位同事同时收到报警信息，并且都尝试去解决这些问题。此时，如果系统内部没有一种通知机制，则大家相互之间并不知道谁在处理哪台主机的问题，更不知道哪些主机的问题还没有人去处理，而且还可能会出现多位同事同时去处理某台主机上的问题的情况。这样，不但浪费人力、物力，而且也不利于故障的及时排除。而引入事件的确认状态后，上述这类问题就可以得到很好的解决。我们知道，当接收到系统发出的报警信息时，就意味着有触发器被触发，相应的也就意味着在系统中有相应的事件产生。这样，不管多少位同事同时处理主机故障，只要大家在动手处理主机故障之前，先确认一下该主机上的问题所引发的事件即可。此时，当有其他同事登录到系统上，准备处理同一台主机问题时，他首先会看到相应的事件已经被确认过了，并且通过查看事件的确认信息，还能了解到是哪位同事在处理这台主机的问题。这样，就可以避免多位同事争着处理同一台主机问题，而留下其他主机的问题没有人去处理的尴尬情况发生。

对事件进行确认的操作非常简单，依次选择菜单项“状态统计”→“事件”；在事件列表页面上找到需要确认的事件，然后单击“确认”列上的“No”（如果对应事件已经被确认过了，则该列会显示“Yes”）超链接，则系统会打开如图 6-28 所示的事件确认页面。

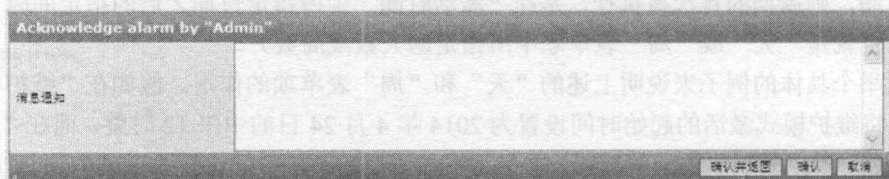


图 6-28 事件确认表单页面截图

由图 6-28 可以看出，事件确认页面的表单内容非常简单，只需在“消息通知 (Message)”文本框里，输入我们所需要确认的信息内容后，单击“确认并返回 (Save and return)”或“确认 (Save)”按钮，即可完成对事件的确认。当单击“确认并返回”按钮时，系统将会在保存确认信息之后将我们带回到事件确认前的页面；而如果单击“确认”按钮，则系统在保存完成确认信息之后，打开的仍然是事件确认页面。

需要注意的是，在 Zabbix 系统中，事件是可以被多次确认的。当你在确认一个事件的时候，如果该事件之前已经被确认过了，则之前被确认的信息将会在事件确认表单页面上，按照确认的先后顺序显示在“消息通知”文本框的上面。另一个需要注意的地方是，当在“常规”页面上配置了不开启事件确认功能时，系统在事件列表页面上就不会显示“确认”列，我们当然也就不能针对事件进行确认了。事件的确认状态可以通过“状态统计”→“状态面板”、“状态统计”→“事件”以及“状态统计”→“触发器”等页面查看到，并且通过这些页面，都可以打开事件确认表单页面对事件进行确认。

6.8.4 导出与导入

在某些情况下，我们可能需要同时管理多台 Zabbix 服务器，且这些 Zabbix 服务器之间是相互独立的。如果这些 Zabbix 服务器所监控的主机的种类甚至型号大部分都是相同的。此时，如果在这些 Zabbix 服务器上分别为所要监控的主机创建模板、自动发现规则等，则不但费时费力、烦琐，而且容易出错。但是，如果在一台 Zabbix 服务器上配置好模板、自动发现规则等，然后导出来，之后将导出来的文件在其他的 Zabbix 服务器上导入。这样，操作起来不但高效而且不容易出错。Zabbix 系统中的导出与导入功能，就是用于完成这类工作的。实际上，使用 Zabbix 系统所提供的导出与导入功能，不但可以提升我们的工作效率，而且还可以用它来相互分享监控配置、交流学习经验等。

在 Zabbix 系统中，支持使用两种方式导出或导入监控元素的配置。第一种方式是，通过 Zabbix 系统的 Web 前端组件导出或导入配置。这种方式操作简单而且方便，也是我们经常使用的导入导出方式。另一种方式是，通过 Zabbix 系统所提供的 API 接口导出导入监控元素的配置，这种方式需要额外编写程序。通过 Web 平台导入或导出的配置信息，只能是 XML 格式的文件，而通过 API 方式，则所导入或导出的配置信息可以是 XML 或 JSON 格式的。

通过 Zabbix 系统的导出与导入功能，可以将以下这些监控元素的配置信息从 Zabbix 系统中导出或将其导入到 Zabbix 系统中：

- ❑ 主机组的配置（要导入或导出主机组的配置，则只能通过 API 方式）。
- ❑ 模板（包括直接在被导出模板上配置的监控项目、触发器、数据图、图表、低级自动发现规则以及模板的关联信息等）。
- ❑ 主机（包括直接在被导出主机上配置的监控项目、触发器、数据图、低级自动发现规则和它所关联的模板信息等）。
- ❑ 网络拓扑图（包括与拓扑图相关的图片等）。
- ❑ 图片。但是如果要想通过 Web 前端组件导出或导入图片，则只能通过导出或导入拓扑图的方法，将引用了要导出或导入的图片所对应的拓扑图导出或导入系统，即可将指定的图片导出或导入系统。不过在导入系统时，可以选择只导入图片而不导入拓扑图。而且，当一张图片被导出或导入系统时，系统并不仅仅是导入或导出了这张图片在系统中的配置信息，同时也会将对应图片的文件内容也导出或导入系统。换句话说，如果从一个 Zabbix 系统中导出了一张图片，并将它导入到另外一个 Zabbix 系统中，则我们不需要在被导入的系统中重新上传图片，就可以看到被导入图片的具体内容了。
- ❑ 图表。

通过 Web 前端组件导出或导入配置信息的操作方法和操作过程比较简单。当导出某种类型监控元素的配置信息时，只需打开对应类型元素的列表页面，然后选中需要导出的对象，并从列表页面的左下侧的下拉列表里选择“导出选中项（Export selected）”选项，最后单击“确认”按钮，就可以将所选中对象的配置信息导出成一个 XML 格式的文件。例如，如果要导出系统中某些模板的配置信息，则只需依次选择菜单项“系统配置”→“模板”，然后在模板的列表页面上选中需要导出的模板，并从该页面上的左下角的下拉列表中选择“导出选中项”，最后单击“确认”按钮，根据系统提示选择要保存的文件路径，就可以将选中模板的配置信息导出到本地的一个 XML 文件中。而要导出系统中主机的配置信息，操作方法与导出模板的配置信息是类似的，只是要在主机列表页面上选中要导出的主机。导入模板或主机配置信息的操作也很简单，只需在模板或主机列表页面上单击“导入”按钮，然后根据系统的提示选择要导入的

配置文件，即可导入相应的配置信息。监控元素配置信息的导出和导入操作比较简单，但是在导出配置信息时需要注意以下几点：

- 所有被选中的监控元素的配置信息将会被导出到同一个文件中。例如，当一次选中多个模板时，则这些模板，包括在这些模板上所配置的监控项目、触发器、数据图、低级自动发现规则等配置信息都会被导出到同一个文件中。
- 当导出主机的配置信息时，主机从所关联的模板上继承过来的配置信息，如监控项目、触发器、数据图以及低级自动发现规则等，将不会一起被导出。类似的，当导出模板的配置信息时，如果所导出的模板关联了其他模板，则被导出模板从所关联模板上继承过来的配置信息，如监控项目、触发器、数据图以及低级自动发现规则等，也将不会随着模板的配置信息一起被导出。虽然主机和模板从所关联的模板那里继承过来的监控元素的配置信息，在主机或模板的配置信息被导出时不会随着主机或模板的配置信息一同被导出；但是，如果导出的文件在另一个 Zabbix 系统中被导入，而且该系统中也存在相应的被关联的模板，则这些未被导出的监控元素会在目标系统中被重建。
- 在导出配置信息时，系统依据低级自动发现规则在被导出监控元素上自动创建的实例，例如，系统依据一个低级自动发现规则，在一台主机上自动创建的监控项目、触发器、数据图等，其配置信息也不会被导出，而仅导出自动发现规则、监控项目样板、触发器样板和数据图样板等的配置信息。
- 如果引用了 Web 监控项目的触发器和数据图等配置信息，则在系统导出主机或模板的配置信息时不会被导出。

类似的，当向一个 Zabbix 系统中导入配置信息时 also 需要注意以下几点：

- 在导入过程中，一旦出现错误，则整个导入过程都将停止。
- 如果导入的内容是图片，且被导入的图片在被导入的系统中是存在的（其实就是在被导入的图片名称在被导入的系统中存在，而不是指图片文件在目标系统中存在），则被导入图片的类型被设置成目标系统中同名图片所对应的类型。例如，被导入的图片在目标系统中是已存在的“小图片”，则即使新导入的图片在源系统中（或者说在被导入的配置文件中）被定义成“背景”，该图片在被导入到新的系统中后将会变成“小图片”，但是图片的实际内容将会被修改成被导入图片的内容。
- 如果在被导入的配置文件中，定义监控项目、触发器、数据图、主机或模板的分类、自动发现规则、监控项目样板、触发器样板、数据图样板等的某些配置项是空的（也就是没有内容），则这些配置项在导入到目标系统中后也将是空的；而对于某些配置项，例如监控项目的分类等，如果是空的，则意味着不覆盖目标系统中同一配置项目的配置。
- 被导入的配置文件可以是 XML 或 JSON 格式的配置文件，但是，配置文件的格式要与文件的扩展名相对应。XML 格式配置文件的扩展名必须是.xml；而 JSON 格式配置文件的扩展名则需要是.json。

6.8.5 全局搜索

当监控系统中定义的监控元素很多时，通过全局搜索（Global search）功能快速查找所需的元素是一个很不错的方法。在 Zabbix 系统的 Web 前端组件的每个页面的右上角，几乎都有一个搜索表单，它就是全局搜索表单。通过这个全局搜索表单，可以快速地在系统中查找主机、

主机组和模板等信息。

当在全局搜索表单中输入要搜索的内容时，系统会显示一个下拉列表，如图 6-29 所示。

全局搜索表单的下拉列表中显示的内容是系统从数据库中查找的，以我们所输入的内容开头的所有主机的列表。当主机名和主机的显示名称都被配置了内容时，系统在搜索的时候匹配的是主机的“显示名称”中的内容；而如果主机的“显示名称”没有被配置内容，则系统会自动匹配“主机名称”中的内容。

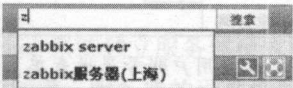


图 6-29 全局搜索表单截图

Zabbix 系统中的全局搜索功能可以用于搜索主机、模板和主机组等监控元素的信息。当搜索主机时，系统将会以主机的显示名称或主机名（当主机的显示名称没有被定义时，系统将使用主机名代替主机的显示名称）、IP 地址和主机的 DNS 主机名作为搜索目标对象；而当搜索模板时，则系统将以模板的名称作为搜索的目标对象；而当搜索主机组时，系统同样以主机组的名称作为搜索的目标对象。

全局搜索的搜索结果将按主机、主机组和模板分区块显示，如图 6-30 所示。

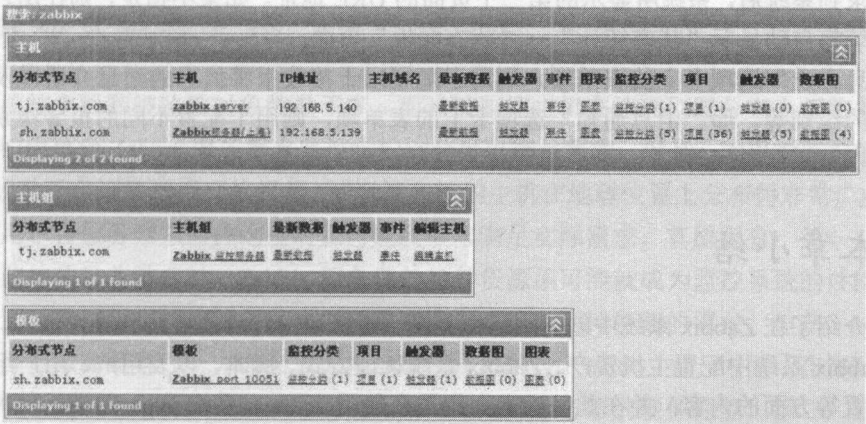


图 6-30 全局搜索结果截图

从图 6-30 可以看出，不仅搜索结果按主机、主机组和模板分区块显示，而且如果单击每条结果各字段上的超链接，则可以查看相关数据或修改相关配置信息。例如，单击主机中的“最新数据”超链接，可以查看对应主机所采集到的最新监控数据；而单击主机名称上的超链接，则可以对对应主机的配置信息进行修改。但是，需要注意的是，如果用户对搜索结果中的对象没有写权限，则系统将以灰色字体显示需要写权限的对应字段的内容。

6.8.6 配置账号属性

与全局搜索功能类似，在 Zabbix 系统 Web 前端组件的每个页面的右上角，几乎都有一个“配置 (profile)”超链接（当以 Guest 用户的身份登录系统时除外），单击这个超链接，每个用户都可以修改自己的账号属性，如图 6-31 所示。

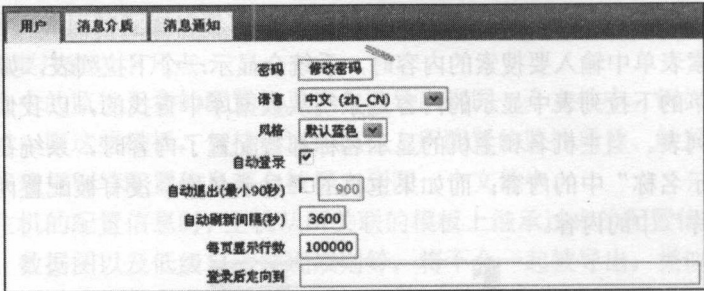


图 6-31 配置用户账号属性表单页面截图

从图 6-31 可以看出，修改用户账号属性的表单页面上有三个选项卡：“用户（User）”、“消息介质（Media）”和“消息通知（Messaging）”等。其中，当用户的账号类型属于普通用户时，图 6-31 所示的修改用户账号属性表单页面上将没有“消息介质”选项卡。

其中，在“用户”选项卡上，可以修改用户账号的密码、页面显示语言、页面显示风格等。若勾选“自动登录（Auto-login）”复选框，则当用户第一次登录到系统时，系统会记住用户密码，以后用户重新进入系统时不需要重新登录。而“每页显示行数（Rows per page）”表单项，则用于配置列表页面上最多显示多少条记录。表单项“登录后定向到（URL）”，则用于指定用户在登录到系统后，系统所显示的第一个页面的 URL 地址。如果不指定，则在用户登录到系统后，系统将自动显示“状态统计”→“状态面板”页面。

“消息介质”选项卡上的表单项，则用于配置用于接收报警信息的信息介质以及发送的目标和报警的级别等。而“消息通知”选项卡上的表单项，则用于配置不同的报警级别在页面播放的声音以及播放时间等。

6.9 本章小结

本章介绍了在 Zabbix 系统中创建网络拓扑图、图表和幻灯片的方法和操作过程，并介绍了如何在 Zabbix 系统中配置主机资产、Zabbix 系统认证方式、脚本，以及用户、用户组、IT 服务和常规设置等方面的内容；并在此基础上介绍了在使用 Zabbix 系统 Web 前端组件时可能经常使用到的操作方法和技巧，如批量更新、维护模式配置、导出与导入以及全局搜索等。

第 7 章 分布式监控

在前面章节中陆续介绍了 Zabbix 系统独立服务器模式的配置以及该模式下的 Web 前端组件的使用和操作方法。然而，我们知道，服务器的硬件性能不可能无限地提升，所以，当被监控主机和监控项目的数量达到一定的数量级别时，特别是当被监控主机在地理位置上分布很广，而且跨度很大时，如果还使用独立服务器模式来监控这些在地理位置上分布很广，而数量又很多的被监控主机时就很容易遇到性能瓶颈。而一旦 Zabbix 系统遇到了很大的性能瓶颈，就会给系统的可靠性、稳定性带来很大的问题。比如，如果 Zabbix 服务器的 CPU 负载一直非常高，监控项目的数据图就会经常性产生断图现象，相应的，系统的误报率也会大大地增加。为了解决此类问题，本章将介绍 Zabbix 系统支持的两种分布式解决方案以及它们的配置和管理方法。

7.1 分布式监控介绍

当被监控主机和被监控对象数量不多，而且被监控主机在地理位置分布上都集中在某个地区时，我们用单台 Zabbix 服务器来监控它们完全可以满足需求。但是，如果被监控的主机比较多，比如说数千台服务器或交换机等，或者被监控的主机在地理位置上分布得非常广泛，此时，如果还只用单台服务器来监控这些对象可能就很难满足实际需求。其原因有：第一，当被监控的主机和监控项目非常多时，Zabbix 服务器的硬件资源很可能就成为监控系统的性能瓶颈。当然，可以通过购买更高档的服务器来尽可能地满足系统对硬件资源的需求。但是，一味地追求服务器硬件资源高配置，一方面会大大增加硬件资源的投入成本；另一方面，在实际情况中硬件资源的性能提升也是有限度的，不可能无限制地提升服务器的硬件性能。第二，当被监控主机在地理位置上分布很广，那么就意味着被监控主机与 Zabbix 服务器之间的通信要通过长距离的广域网传输。而通过长距离的广域网传输数据经常会因为网络连接原因导致数据丢失或延时很大，这样就很难满足我们对于监控系统的稳定性、可靠性和及时性的要求。第三，在实际工作中，我们经常会遇到，跨广域网里的两个或多个主机因为多种原因而无法直接通信。比如，没有足够的公网 IP 地址，所以，对于一些内部系统我们经常只配置私网 IP 地址，而不配置公网 IP 地址。这个时候，如果仍然使用单台 Zabbix 服务器，可能就很难对这类主机进行监控。第四，出于安全方面考虑，我们在实际系统运维工作中，都尽可能地对公网用户关闭非必需的服务端口。而如果使用单台 Zabbix 服务器监控分布在多个地理位置上的主机，就必须对公网（至少是对 Zabbix 服务器）开放一些必需的服务端口，比如被监控设备代理服务端口 10050、SNMP 服务的端口 161 等，这样就增加了系统的安全风险。

或许有读者会说，上面所说的这些问题都很好解决，只需在被监控主机本地搭建一套 Zabbix 服务器，以使每台 Zabbix 服务器只监控它本地的被监控主机，不就可以很好地解决了吗？嗯，

你说的没错，如果在每个需要监控系统的地方，都搭建一套 Zabbix 系统是可以某种程度上解决上面所述的这些问题的。但是，如果每个地方的 Zabbix 服务器都是独立的，相互之间没有任何关系，则当 Zabbix 系统的数量很多时，日常管理就是一个很大的问题。比如说，对于同一种类型的被监控主机，需要分别在各个地方的 Zabbix 服务器上创建相应模板等。

既然如此，那你可能会问，那么我们怎样解决上面所述的这些问题呢？其实，解决上述这些问题的方法很简单，请读者不要忘记了，我们在本书的开篇第 1 章介绍 Zabbix 系统时，就提到了，Zabbix 系统是一种企业级的开源的分布式监控系统。既然是分布式的监控系统，那么如果启用 Zabbix 系统的分布式模式，则上面所述的所有问题不就迎刃而解了吗？

针对分布式工作模式，Zabbix 系统提供了两种解决方法。第一种解决方案是：使用单台独立服务器模式工作的 Zabbix 服务器，加上若干台 Zabbix 服务器代理端组成的分布式监控系统；第二种解决方案是：使用一台分布式模式工作的主（Master）Zabbix 服务器和若干台多级的分布式模式工作的从（slave）服务器，加上若干台 Zabbix 服务器代理端组成的分布式监控系统。

那么，Zabbix 系统的这两种分布式解决方案都有什么优缺点呢？在实际工作中，我们该如何从这两种解决方案中，选择一种适合于我们实际需要的解决方案呢？首先，需要说明的是，Zabbix 系统所提供的这两种分布式解决方案之间并不互相排斥。也就是说，Zabbix 系统所提供的这两种分布式解决方案，并不是我们选择了其中一种就不能再选择另外一种。实际上，第二种解决方案可以是以第一种解决方案为基础所进行的扩展。但是，如果因此就说第二种解决方案比第一种解决方案更高级更先进，那也不完全正确。因为，对于第二种解决方案而言，它的分布式节点服务器，也可以没有属于它的 Zabbix 服务器代理节点。至于说这两种解决方案各自的优缺点，我们不能一概而论。一般而言，如果被监控的主机和被监控的监控项目不是特别的多，且被监控主机在地理位置上分布不是特别的广，同时日常负责管理 Zabbix 系统的人员不是很多，特别是没有不同组织的人员需要同时管理监控系统时，使用第一种分布式解决方案比较合适。而如果被监控的主机和被监控的监控项目不是特别多，且负责日常管理监控系统的人员也很多，特别是一个组织的内部有多个子组织时，我们认为选择第二种分布式解决方案可能比较合适。例如，一个集团公司下面有很多子公司，而每个子公司都有自己的运维或 IT 人员需要管理属于本公司内部的主机，这时，我们觉得选择第二种分布式解决方案就可以做到分工和责任明确，比较容易避免管理上的混乱。因为，在这种情况下，使用 Zabbix 系统的第二种分布式解决方案就可以做到，集团公司的运维或 IT 人员负责集团公司内部的骨干网络设备和直属集团公司的核心系统主机的监控，而子公司的运维或 IT 人员只负责属于本公司网络设备和系统主机的监控，因此，在这类场景下，我们觉得适宜选择第二种分布式解决方案。

7.2 单级分布式监控

所谓单级分布式监控，也就是一般使用单台独立服务器模式工作的 Zabbix 服务器加上若干台 Zabbix 服务器代理端组建而成的分布式监控系统，它实际上也就是我们在前面所提到的，Zabbix 系统所提供的两种分布式解决方案中的第一种解决方案。Zabbix 系统的这种分布式解决方案，实施起来很简单，且能做到数据集中管理，集中配置。这种解决方案适合于被监视主机和监控项目的数量不是特别多，而被监控主机在地理位置分布上也不是特别广，且组织结构比

较简单，Zabbix 系统的管理人员不是特别多的情况下使用。

7.2.1 Zabbix 服务器代理组件

Zabbix 服务器代理组件，顾名思义，它是为了减轻 Zabbix 服务器的压力，用以代替 Zabbix 服务器采集监控项目的监控数据，并将所采集到的监控数据定期发给 Zabbix 服务器的 Zabbix 系统组件。因而，使用 Zabbix 服务器代理组件不但可以减轻 Zabbix 服务器采集监控数据的压力，提升 Zabbix 服务器的性能，而且当被监控主机与 Zabbix 服务器处在不同的地理位置上时，使用服务器代理组件还可以大大减小因为网络传输的原因而引起的数据丢失率，从而大大提升监控系统的稳定性、可靠性，并大大减少由此而产生的误报。因为，如果 Zabbix 服务器代理在给 Zabbix 服务器发送所采集到的监控数据时出错，则 Zabbix 服务器代理会不断重试，直到将需要发送的数据发送成功为止。

很显然，为了暂存采集到的监控数据，Zabbix 服务器代理组件也需要有数据库支持，而且在一套 Zabbix 系统中，Zabbix 服务器代理组件所使用的数据库类型，不一定非要和 Zabbix 服务器所使用的数据库类型保持一致。虽然 Zabbix 服务器代理组件需要数据库支持，但是，不可以直接通过它配置被监控主机和监控项目，也不可以直接通过它查看监控项目的数据图或发送报警信息。因为 Zabbix 服务器代理组件没有用户图形界面（GUI），它所监控的主机和监控项目的配置信息都是定期从 Zabbix 服务器那里获取的。因此，从这个角度来说，Zabbix 服务器代理组件实际上只是一个监控数据的采集器和收集器，而不可以通过它对所采集和收集的监控数据做进一步的处理。具体来说，通过 Zabbix 服务器代理组件，可以实现以下功能：

- 采集和收集远程被监控主机上的监控数据。
- 采集和收集本地与 Zabbix 服务器之间没有可靠通信的主机的监控数据。
- 当 Zabbix 系统所监控的被监控主机的数量比较多时，通过 Zabbix 服务器代理可以减轻 Zabbix 服务器的负载压力，提升监控系统的性能，使 Zabbix 服务器从数据采集的繁重工作之中解放出来，从而专职于数据处理与报警信息的发送等。
- 减化分布式监控系统维护的操作过程。

具体来说，利用 Zabbix 服务器代理组件可以完成的功能如表 7-1 和表 7-2 所示。

表 7-1 Zabbix服务器代理组件所支持的监控数据采集方法列表

监控数据采集方法	是否支持
通过被监控设备代理（agent）采集监控数据（包括主动和被动方式）	支持
简单检查	支持
通过SNMP协议采集监控数据（包括SNMP陷入）	支持
Zabbix系统内部数据采集	不支持
Zabbix陷入	支持
数据聚合	不支持
通过外部脚本采集监控数据	支持
数据库监控	不支持
通过IPMI代理采集监控数据	支持
通过SSH协议采集监控数据	支持

续表

监控数据采集方法	是否支持
通过TELNET协议采集监控数据	支持
通过JMX协议采集监控数据	支持
通过计算的方法采集监控数据	不支持

表 7-2 Zabbix服务器代理组件所支持的其他功能列表

功能	是否支持
Web监控	支持
网络自动发现	支持
低级自动发现	支持
触发器表达式的计算	不支持
处理事件	不支持
发送报警信息	不支持
执行远程命令	不支持

既然 Zabbix 服务器代理要将其所采集的监控数据定时地发送给 Zabbix 服务器进行处理，而且还要定期从 Zabbix 服务器那里获取被监控主机和监控项目配置信息的更新，那么，Zabbix 服务器代理和 Zabbix 服务器之间就必须要进行通信。Zabbix 服务器代理与 Zabbix 服务器之间的通信使用的是被广泛使用的 TCP 协议，它们之间的网络拓扑关系如图 7-1 所示。

由图 7-1 可以看出，Zabbix 服务器与 Zabbix 服务器代理可以处在不同的局域网内，然后它们之间通过公网相互连接以便传输监控数据和配置信息。当然，使用 Zabbix 服务器代理组件与 Zabbix 服务器组成的分布式监控系统时，并不一定要求所有的监控项目和被监控主机都通过 Zabbix 服务器代理组件采集监控数据。对于 Zabbix 服务器本地的被监控主机和监控项目的监控数据，仍然可以通过 Zabbix 服务器来采集。也就是说，对于监控数据的采集，Zabbix 服务器和 Zabbix 服务器代理之间并不存在排斥关系，例如图 7-1 所示，与 Zabbix 服务器同处于一个局域网里的主机，仍然可以通过 Zabbix 服务器来采集监控数据。

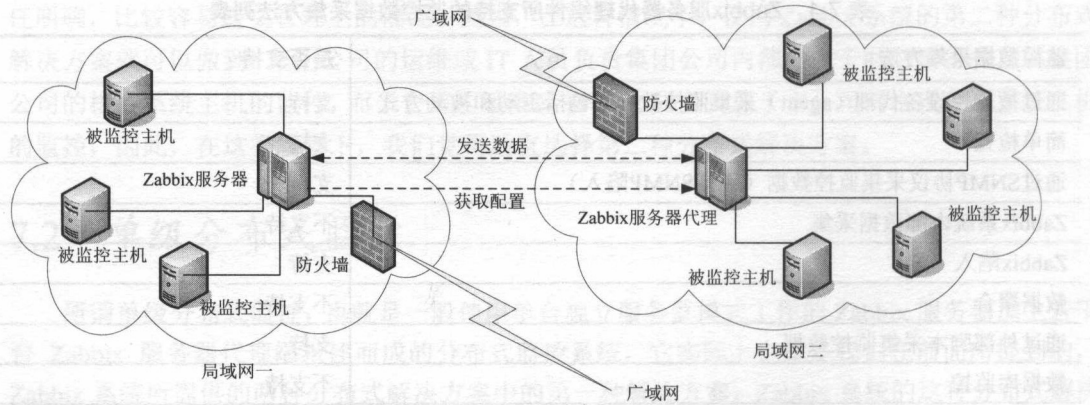


图 7-1 Zabbix 服务器与 Zabbix 服务器代理之间关系拓扑图

7.2.2 Zabbix 服务器代理组件安装

Zabbix 服务器代理组件的安装过程与 Zabbix 服务器的安装过程类似，只是因为 Zabbix 服务器代理组件没有用户图形化界面，所以不需要安装 Web 服务器环境，且不需要安装部署 Zabbix 系统的 Web 前端组件。接下来，我们简单地介绍一下 Zabbix 服务器代理组件的安装部署过程。

1. 安装 MySQL 数据库

如前文所述，因为 Zabbix 服务器代理组件需要数据库支持，所以，首先当然需要在服务器上安装数据库。与 Zabbix 服务器一样，Zabbix 服务器代理组件支持 MySQL、Oracle 及 Postgre 等众多关系型数据库。我们这里选择 MySQL 数据库作为 Zabbix 服务器代理组件的数据库，所以，接下来就需要安装 MySQL 数据库。MySQL 数据库的具体安装步骤和安装命令，请读者参阅我们在第 1 章中介绍 Zabbix 独立服务器安装时的步骤，我们在此不再一一复述其具体的安装过程和安装命令。

2. 安装 Java 环境

如果要想 Zabbix 服务器代理组件能够通过 JMX 协议采集 Java 应用或 Java 容器的监控数据，则需要在 Zabbix 代理服务器上安装 JDK 环境。安装 JDK 环境比较简单，但是需要提请注意的是，操作系统的位数要和所安装的 JDK 软件包的位数相一致（即要么都是 32 位的，要么都是 64 位的）。以下是安装 JDK 环境的简单安装步骤：

```
shell> chmod +x jdk-6u38-linux-i586.bin           #设置软件包可执行权限
shell> ./jdk-6u38-linux-i586.bin                  #执行软件包，让其自解压
shell> mv jdk1.6.0_38 /usr/java                    #将解压好的目录移到正式目录下
#设置 Java_HOME 环境变量
shell> echo "Java_HOME=/usr/java" >>/etc/profile
#设置 PATH 环境变量
shell> echo "PATH=\$PATH:/usr/java/bin" >>/etc/profile
shell> echo "export Java_HOME" >>/etc/profile
shell> echo "export PATH " >>/etc/profile
shell> source /etc/profile                          #使环境变量生效
```

3. 编译安装 libssh 软件包

libssh2-1.4.3 是 Zabbix 服务器代理组件支持通过 SSH 协议来采集监控数据所必需的软件包，如果打算通过 SSH 协议采集监控数据，则在安装 Zabbix 服务器代理组件之前需要安装这个软件。否则，可以跳过这个安装步骤，直接执行下一步安装。

```
shell> tar -zxvf libssh2-1.4.3.tar.gz              #解压软件包
shell> cd libssh2-1.4.3                            #切换到源码包的根目录
# 执行编译配置脚本，以进行编译前的编译配置
shell> ./configure
shell> make && make install                          # 编译并安装软件包
```

4. 编译安装 OpenIPMI-2.0.20-rc1 软件包

如果打算让 Zabbix 服务器代理组件能够通过 IPMI 协议采集监控数据，则就必须要安装 OpenIPMI 软件包，否则可以跳过本步骤。以下是安装 OpenIPMI 软件包的简单步骤：

```
shell> tar -zxvf OpenIPMI-2.0.20-rc1.tar.gz        #解压软件包
shell> cd OpenIPMI-2.0.20-rc1                      #切换到源码包的根目录
# 执行编译配置脚本，以进行编译前的编译配置
shell> ./configure
shell> make && make install                          # 编译并安装软件包
```


5. 编译安装 Zabbix 服务器代理组件

相信通过 SNMP 协议来采集监控数据是大家都需要使用的一种监控数据采集方法，所以，在编译安装 Zabbix 服务器代理组件之前，请通过下列命令（Redhat 或 Centos 操作系统）检查一下你的系统上是否安装了 net-snmp、net-snmp-devel、net-snmp-libs 和 net-snmp-utils 等软件包。

```
shell> rpm -qa |grep net-snmp
```

如果系统上没有安装上述软件包，或者只安装了上述的其中几个软件包，则请你在编译安装 Zabbix 服务器代理组件之前，通过 yum 或 rpm 命令将上述的 4 个软件包都安装到系统中。

类似的，如果打算让 Zabbix 服务器代理支持 Web 项目的监控，则请在编译安装 Zabbix 服务器代理组件之前，使用下列命令检查你的系统中是否安装了 libcurl 和 libcurl-devel 软件包。

```
shell> rpm -qa |grep libcurl
```

如果经检查你的系统中没有安装上述这两个软件包，则请你在编译安装 Zabbix 服务器代理组件之前，使用 yum 或 rpm 命令（Redhat 或 Centos 系统）安装好上述两个软件包。

从安全的角度出发，一般 Zabbix 服务器代理进程使用非 root 用户运行，所以需要在系统上创建一个用户账号和一个用户组，以用于运行 Zabbix 服务器代理组件。请用下列命令在系统中创建用户和用户组。

```
shell> useradd zabbix
```

6. 创建 Zabbix 用户和用户组

在 Redhat、Centos 等发行版的 Linux 系统上，使用上述 useradd 命令创建用户账号时，系统会自动创建一个对应的用户组，且用户组名与所创建的账号名相同。所以，我们在这里不需要另外创建用户组。

好了，经过上述的准备，接下来，就可以通过编译安装的方式编译安装 Zabbix 服务器代理组件了。以下是编译安装 Zabbix 服务器代理组件的简单步骤。

```
shell> tar -zxvf zabbix-2.0.0.tar.gz          # 解压软件包
shell> cd zabbix-2.0.0                      # 切换到源码包的根目录
# 执行编译配置脚本，以进行编译前的编译配置
shell> ./configure --prefix=/opt/zabbix \
--enable-proxy --enable-agent \
--enable-java --with-mysql=/opt/mysql/bin/mysql_config \
--with-net-snmp --with-libcurl \
--with-ssh2 --with-openipmi
```

上述执行编译配置脚本时所输入的配置选项，你可以根据自己的实际需要进行调整。例如，如果不打算让 Zabbix 服务器代理组件通过 JMX 协议采集监控数据，则--enable-java 选项就可以不要；类似的，如果不打算让 Zabbix 服务器代理组件通过 SSH 协议采集监控数据，或者不打算让 Zabbix 服务器代理组件执行 Web 监控，则，--with-ssh2 或--with-libcurl 配置选项也可以不要。

```
shell> make                                # 执行 make 命令进行编译
shell> make install                        # 安装编译好的文件到指定的目录
```

7.2.3 Zabbix 服务器代理组件运行环境配置

如果上述命令都执行成功，则 Zabbix 服务器代理组件就被安装在了/opt/zabbix 目录下。接下来，在启动 Zabbix 服务器代理进程之前，需要对 Zabbix 服务器组件的运行环境做一些简单的配置。

1. 编译安装 fping 工具包

与 Zabbix 服务器一样, Zabbix 服务器代理组件也需要通过调用 fping 命令, 来获取被监控主机到 Zabbix 服务器代理组件之间的 ICMP 信息, 从而判断被监控主机的存活状态。因此, 也需要在 Zabbix 服务器代理组件上安装 fping 工具包。fping 工具包的安装过程和步骤如下所示:

```
shell> tar -zxvf fping-2.4b2_to.tar.gz          # 解压软件包
shell> cd fping-2.4b2_to                      # 切换到源码包的根目录
# 执行编译配置脚本, 以进行编译前的编译配置
shell> ./configure
shell> make && make install                    # 编译并安装软件包
```

2. JMX 环境配置

同样, 如果要让 Zabbix 服务器代理能够通过 JMX 协议采集监控数据, 则需要在 Zabbix 服务器代理上配置 JMX 数据采集环境。

在配置 JMX 环境之前, 首先需要确保已经安装了 Zabbix Java 应用程序网关程序。如果在编译安装 Zabbix 服务器代理组件时使用了 --enable-java 编译配置选项, 则说明已经选择编译安装 Zabbix Java 应用程序网关了。如果选择了编译安装 Zabbix Java 应用程序网关, 则在 Zabbix 安装目录下的 sbin 目录下会看到 zabbix_java 目录, 且有相应文件。

在 Zabbix 服务器代理组件的配置文件 etc/zabbix_proxy.conf 中, 涉及与 JMX 协议相关的配置项主要有三个。它们是: JavaGateway=IP/hostname, 用于指定运行 Zabbix Java 应用程序网关进程的服务器 IP 地址或主机, 一般是本机, 因此配置成 127.0.0.1 即可; JavaGatewayPort=Port, 用于指定连接 Zabbix Java 应用程序网关的端口, 默认为 10052; StartJavaPollers=2, 设定启动多少个进程与 Zabbix Java 应用程序网关通信。

Zabbix Java 应用程序网关的配置文件为 Zabbix 安装目录下的 sbin/zabbix_java/settings.sh。这个配置文件里的内容很简单: LISTEN_IP="0.0.0.0"配置项, 用于指定应用程序网关所侦听的 IP 地址, 如果配置为 0.0.0.0, 则表示侦听本机上所有网卡上所配置的 IP 地址; LISTEN_PORT=10052 配置项, 用于指定侦听的端口号, 默认端口号为 10052; 配置项 PID_FILE="/tmp/zabbix_java.pid", 用于指定启动 Zabbix Java 应用程序网关后, 应用程序网关所生成的 PID 文件及路径; START_POLLERS=5 配置项, 则用于设定启动多少个进程来具体负责监控数据的采集。需要说明的是, START_POLLERS 配置项所指定的数字, 不应小于 Zabbix 服务器代理组件配置文件中, StartJavaPollers 配置项所指定的数字。否则, 可能会造成 Zabbix 服务器端进程无法连接 Zabbix Java 应用程序网关, 而造成监控数据丢失的情况发生。

完成上述配置文件的修改后, 就可以使用 sbin/zabbix_java/startup.sh 命令, 来启动 Zabbix Java 应用程序网关进程了。关闭 Zabbix Java 应用程序网关进程的命令是 sbin/zabbix_java/shutdown.sh。

3. 初始化系统数据库及配置服务器代理组件

与 Zabbix 服务器端组件一样, Zabbix 服务器代理组件也需要数据库支持, 所以在启动 Zabbix 服务器代理进程之前, 需要对它需要使用的数据库进行初始化。初始化的操作步骤和命令如下所述:

```
#通过 mysql 客户端工具连接到 MySQL 服务
shell> mysql -uroot -ppassword
# 创建 Zabbix 服务器代理组件需要使用的数据库 zabbix_proxy, 设置数据库的默认编码为 utf8。
mysql> CREATE DATABASE `zabbix_proxy` /*!40100 DEFAULT CHARACTER SET utf8 */;
# 创建一个 mysql 用户, 并将 zabbix_proxy 数据库的所有访问权限都赋给这个用户
mysql> grant all on zabbix_proxy.* to 'zabbix_proxy'@'localhost' identified by
```

```
'zabbix_proxy';
# 将 Zabbix 软件包中的 SQL 语句导入进 zabbix_proxy 数据库
shell> mysql -uzabbix_proxy -pzabbix_proxy zabbix_proxy <schema.sql
# 导入初始的图片信息到 zabbix_proxy 数据库
shell> mysql -uzabbix_proxy -pzabbix_proxy zabbix_proxy <images.sql
# 导入初始数据到 zabbix_proxy 数据库
shell> mysql -uzabbix_proxy -pzabbix_proxy zabbix_proxy <data.sql
```

提示：上述数据库数据的导入顺序不可以错，否则相关的数据将无法导入到数据库中。

完成上述数据库的初始化后，接下来，就需要修改 Zabbix 服务器代理组件的配置文件，以使其与我们系统的实际环境相一致。Zabbix 服务器代理组件的修改方法和修改内容如下所述：

```
shell> vi /optt/zabbix/etc/zabbix_proxy.conf
Server=127.0.0.1      => Server=192.168.5.139.
```

该配置项用于指定 Zabbix 服务器代理将要连接的 Zabbix 服务器的 IP 地址或主机名。如果 Zabbix 服务器代理组件工作在主动模式下，则 Zabbix 服务器代理组件将从这个配置项所指定的 Zabbix 服务器上，获取被监控主机和监控项目等监控元素的配置信息。而如果 Zabbix 服务器代理组件工作在被动模式下，则系统将忽略这个配置项中所配置的内容。Zabbix 服务器代理是否工作在主动工作模式下，是通过 ProxyMode 配置项指定的。当这个配置项被设置成 0 时，则 Zabbix 服务器代理将工作在主动工作模式下，否则它将工作在被动工作模式下。

```
Hostname=zabbix proxy  => Hostname=Shanghai proxy.
```

这个配置项用于配置 Zabbix 服务器代理组件的名称。该配置项所配置的名称虽然不需要是 DNS 解析的合法主机名，但是，它必须在整个 Zabbix 监控系统环境中是唯一的，不得有重复，而且其是大小写敏感的。同时，该配置项所指定的 Zabbix 服务器代理组件的名称，要与将来通过 Web 组件页面添加的 Zabbix 服务器代理组件名称保持一致。

```
DBName=zabbix      => DBName=zabbix_proxy.
```

上面配置项用于指定，Zabbix 服务器代理组件将要连接的数据库的数据库名称。

```
DBUser=zabbix      => DBUser=zabbix_proxy.
```

上面配置项用于指定 Zabbix 服务器代理组件连接数据库时所使用的用户名。

```
# DBPassword=      => DBPassword=zabbix_proxy.
```

上面这个配置项用于指定 Zabbix 服务器代理组件连接数据库时所使用的用户密码。当 Zabbix 服务器代理组件连接数据库时所使用的用户账号没有设置密码时，则需要注释掉这个配置项。

```
# StartIPMIPollers=0      => StartIPMIPollers=1.
```

如果系统中有监控项目是通过 IPMI 协议采集数据的，则需要开启这个配置项，且需要根据你的系统中 IPMI 类型监控项目数量的多少，调整这个配置项值的大小。

```
# StartJavaPollers=0      => StartJavaPollers=1.
```

如果系统中有监控项目是通过 JMX 协议（或者你打算让 Zabbix 服务器代理组件通过 JMX 协议采集数据）采集数据的，则需要开启这个配置项，且需要根据系统中 JMX 类型监控项目数量的多少，调整这个配置项值的大小。

```
# StartSNMPTrapper=0      => StartSNMPTrapper=1.
```

如果系统中有 SNMP 陷入类型的监控项目，或者你打算将来可能创建这种类型的监控项目，则需要开启这个配置项，且需要根据系统中 SNMP 陷入类型监控项目数量的多少，调整这个配置项值的大小。

```
# FpingLocation=/usr/sbin/fping  => FpingLocation=/usr/local/sbin/fping
```


上面配置项用于指定 `fping` 命令工具的路径。

完成上述配置之后,使用下列命令启动 Zabbix 服务器代理进程以及 Java 应用程序网关进程。

```
# 启动 Zabbix 服务器代理进程。
shell> /opt/zabbix/sbin/zabbix_proxy -c /opt/zabbix/etc/zabbix_proxy.conf
# 启动 Java 应用程序网关进程。
shell> /opt/zabbix/sbin/zabbix_java/startup.sh
```

7.2.4 Zabbix 服务器代理节点的添加及使用

当安装并配置好 Zabbix 服务器代理组件后,接下来就需要通过 Web 前端页面将所创建的 Zabbix 服务器代理组件添加到 Zabbix 系统中,并设置相应的被监控主机由其进行监控数据的采集与监控。

要在 Zabbix 系统中添加一个 Zabbix 服务器代理节点或修改一个已存在的 Zabbix 服务器代理节点的配置,可依次选择菜单项“高级配置 (Administration)”→“节点管理 (DM)”。系统将打开“配置监控代理”列表页面,该页面默认情况下显示的是系统中已经存在的 Zabbix 服务器代理节点列表。单击页面右上部的“新建代理节点”按钮,或者单击该页面列表中某个 Zabbix 服务器代理节点名称上的超链接,系统都会打开“配置监控代理”的表单页面,如图 7-2 所示。

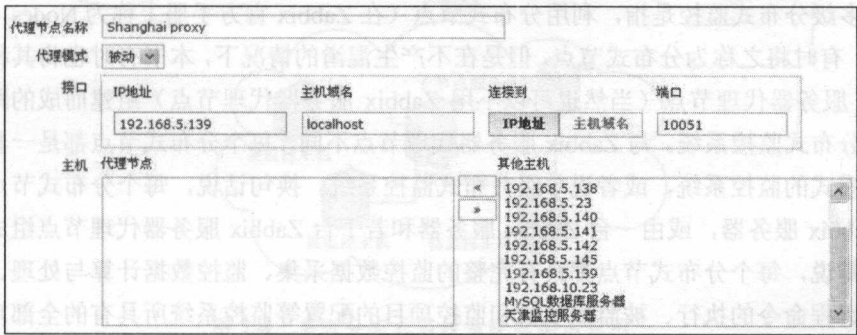


图 7-2 配置监控代理表单页面截图

由图 7-2 可知,“配置监控代理”表单页面上的内容比较简单。其中,“代理节点名称 (Proxy name)”表单项用于配置代理节点的名称,该名称是大小写敏感的,而且要求代理节点名称在系统中是唯一的,不可以重复,同时需要与 Zabbix 服务器代理组件配置文件中 `Hostname` 配置项所配置的内容保持一致;“代理模式 (Proxy mode)”表单项,用于配置对应 Zabbix 服务器代理是工作在主动模式下还是工作在被动模式下;当选择代理模式为被动模式时,系统会显示出附加的“接口”表单域,它用于配置被添加代理节点的 IP 地址或者主机域名;最后的“主机”表单域,用于配置哪些被监控主机通过该 Zabbix 服务器代理节点采集监控数据。

除了在图 7-2 所示的配置 Zabbix 服务器代理节点的表单页面上,指定哪些被监控主机将通过 Zabbix 服务器代理采集监控数据,还可以在配置监控主机时,通过“配置主机”表单页面上的“由代理节点监控 (Monitored by proxy)”表单项,指定对应的主机将通过哪台 Zabbix 服务器代理组件采集监控数据,如图 7-3 所示。



图 7-3 配置主机的代理节点表单项截图

7.3 多级分布式监控

在 7.2 节，介绍了利用 Zabbix 服务器代理组件实现的单级分布式监控系统的安装、部署和配置方法等。之所以称之为单级分布式监控系统，是因为仅使用独立服务器加上若干台 Zabbix 服务器代理组件所组成的分布式监控系统，其监控数据和被监控主机与监控项目的配置信息都是集中存储在一台数据库服务器上的。也就是说，各种数据不是分级存储的，而且通过这种方式实现的分布式监控系统，所有的被监控主机和监控项目的配置都是通过与 Zabbix 服务器相关的那个 Web 前端组件完成操作的；同时报警信息的发送、监控数据的处理（比如说触发器表达式的计算等）以及远程命令的执行等，仍然还是要通过 Zabbix 服务器来完成。而多级分布式监控系统则不同，多级分布式监控系统，不但可以实现监控数据和监控配置信息数据的集中存储，而且还可以实现多级存储；同时，多级分布式监控系统还可以实现被监控主机和监控项目的本地配置以及本地报警，即可以实现多点多地地配置监控信息和查看监控数据。接下来介绍 Zabbix 系统中多级分布式监控系统的安装、部署和配置方法以及日常维护操作流程等。

7.3.1 多级分布式监控的结构

所谓多级分布式监控是指，利用分布式节点（在 Zabbix 官方手册上称为 Nodes，在本书中为了区分，有时将之称为分布式节点，但是在不产生混淆的情况下，本书有时也将其称为节点）和 Zabbix 服务器代理节点（当然也可以不用 Zabbix 服务器代理节点）组建而成的跨区域的、多层次的分布式监控系统。与 Zabbix 服务器代理节点不同，每个分布式节点都是一套完整的独立服务器模式的监控系统，或者说单级分布式监控系统。换句话说，每个分布式节点都是一台完整的 Zabbix 服务器，或由一台 Zabbix 服务器和若干台 Zabbix 服务器代理节点组成的监控系统。也就是说，每个分布式节点都具有完整的监控数据采集、监控数据计算与处理、报警信息的发送与远程命令的执行、被监控主机和监控项目的配置等监控系统所具有的全部功能。而每个分布式节点都只负责自己领域内被监控主机的监控数据的采集、处理与报警等。而不同的分布式节点按层次、按级别组合在一起，就组建成了分层次、分级别的树状结构的多级分布式监控系统。我们先来看看如图 7-4 所示的典型的多级分布式监控系统结构示意图。

从图 7-4 可以看出，整个多级分布式监控系统由多个分布式监控节点组成，这些分布式节点又分为根主节点和子节点。其中，在每套多级分布式监控系统中，至少要有一个根主节点，其他的节点都是这个根主节点的子节点。之所以称之为根主节点，是因为在多级分布式监控系统中，除了根主节点，其他节点既可以是根主节点的子节点，也可以是其他节点的主节点。例如图 7-4 所示的子节点 A，它是根主节点的一个子节点，同时，它又是子节点 B 的主节点。而每个节点（包括根主节点和子节点）又都可以是由单台 Zabbix 服务器加上若干台 Zabbix 服务器代理组成的单级分布式监控系统。当然如果某个节点没有 Zabbix 服务器代理，而完全由 Zabbix 服务器来负责监控数据的采集也是允许的。但是，不管是根主节点还是子节点，它们都只能负责本节点范围内监控对象的监控数据的采集、计算与处理，并实施报警信息的发送等。任何一个节点，包括根主节点不能跨节点对其他节点内监控对象的监控数据进行采集与计算。例如，图 7-4 中“主节点”虽然是子节点 C 的主节点，但是，它不可以对子节点 C 所监控的被监控主机进行监控数据的采集和计算，子节点 C 所管辖范围内的被监控主机的监控数据的采集，只能

由子节点 C 的节点服务器或其 Zabbix 服务器代理节点来完成。

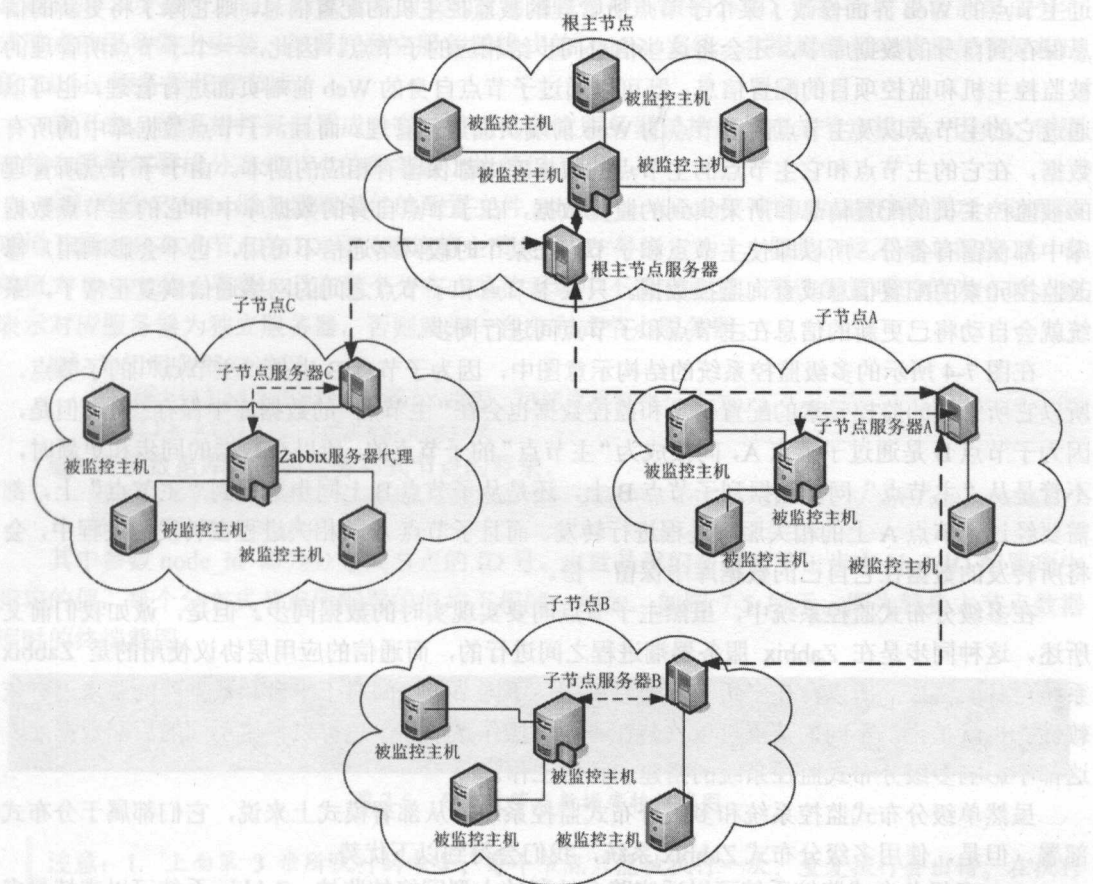


图 7-4 多级分布式监控系统结构示意图

虽然，主节点不能对隶属于它的子节点所管辖的主机进行监控数据的采集与处理，但是，通过主节点的 Web 前端组件，可以管理隶属于它的所有子节点的监控元素，并且可以查看所有隶属于它的所有子节点所采集到的监控数据。例如，通过图 7-4 中的“主节点”的 Web 前端组件，可以管理子节点 A 所监控的所有被监控主机和监控项目的配置信息，并且，可以从“主节点”的 Web 前端页面上查看子节点 A 所采集的所有被监控主机的监控数据以及数据图、图表和拓扑图等（当然用户账号得有相应的查看权限才可以）。虽然子节点 B 是子节点 A 的子节点，而子节点 A 又是“主节点”的子节点；但是，如果没有将子节点 B 添加为“主节点”的子节点（通过 Web 前端组件进行配置，后面我们将详细介绍具体的配置过程和方法），则“主节点”就不可以直接管理子节点 B 上监控元素的配置，也不可以查看子节点 B 上的监控数据和数据图等。

或许读者会问，既然主节点能够管理子节点所管理的主机，并且能够查看子节点所监控主机的监控数据，那么这些功能是通过什么机制实现的呢？难道是主节点的服务器进程在必要的时候，实时操作子节点的数据库吗？不是这样的。实际上，主节点和子节点的相关进程会根据配置，定时同步相互之间的数据。也就是说，子节点所采集的监控数据和它所管理的被监控主机和监控项目的配置信息，除了会在它自身的系统数据库中保存，还会通过相关进程定期地同

步到它的主节点上，并且主节点也会将这些数据保存在自己的数据库中。相应的，如果用户通过主节点的 Web 界面修改了某个子节点所管理的被监控主机的配置信息，则它除了将更新的信息保存到自身的数据库中，还会将这些信息同步给相应的子节点。因此，一个子节点所管理的被监控主机和监控项目的配置信息，既可以通过子节点自身的 Web 前端页面进行管理，也可以通过它的主节点以及主节点的主节点的 Web 前端页面进行管理。而且，子节点数据库中的所有数据，在它的主节点和它主节点的主节点的数据库中都保留有相应的副本。由于子节点所管理的被监控主机的配置信息和所采集到的监控数据，在子节点自身的数据库中和它的主节点数据库中都保留有备份，所以即使主节点和子节点在某个时段网络通信不可用，也不会影响用户修改监控元素的配置信息或查询监控数据。只要主节点和子节点之间的网络通信恢复正常了，系统就会自动将已更新的信息在主节点和子节点间进行同步。

在图 7-4 所示的多级监控系统的结构示意图中，因为子节点 B 也是“主节点”的子节点，所以它所管理的监控元素的配置信息和监控数据也会在“主节点”的数据库中保存一份。但是，因为子节点 B 是通过子节点 A，间接成为“主节点”的子节点的，所以在数据的同步和更新时，不管是从“主节点”同步数据到子节点 B 上，还是从子节点 B 上同步数据到“主节点”上，都需要经过子节点 A 上的相关服务进程进行转发。而且子节点 A 的相关进程在转发的过程中，会将所转发的数据在它自己的数据库中保留一份。

在多级分布式监控系统中，虽然主子节点间要实现实时的数据同步。但是，诚如我们前文所述，这种同步是在 Zabbix 服务器端进程之间进行的，而通信的应用层协议使用的是 Zabbix 系统自身的协议。所以，在多级分布式系统中，不同节点之间对于平台和数据库的要求没有依赖性。也就是说，不同的节点可以安装在不同的操作系统下，也可以使用不同的后端数据库，这都不影响多级分布式监控系统的搭建和正常工作。

虽然单级分布式监控系统 and 多级分布式监控系统，从部署模式上来说，它们都属于分布式部署。但是，使用多级分布式 Zabbix 系统，我们会得到以下优势：

- 多级分布式监控系统可以适应跨多地区的大型网络的监控。Zabbix 系统可以支持最多 999 个分布式节点，而每个分布式节点又都可以是一套单级分布式监控系统，所以，使用多级分布式监控系统可以支持对非常大型的网络进行监控。
- 每个分布式节点，既可以通过其自身的 Web 前端页面管理其本地的监控元素，也可以通过它的主节点和它的主节点的主节点，来配置和管理它所管理的监控元素。
- 本地监控数据的采集将不会因为网络原因而受到影响，即使主节点和子节点之间有通信故障，也不会影响子节点的正常工作。
- 增减子节点不会对现有的任何节点带来任何影响。

注意：当系统的后端数据库使用 SQLite 数据库时，不可以组建多级分布式监控系统。

7.3.2 多级分布式监控系统的安装与部署

安装和部署一套多级分布式监控系统大致需要经过以下几个步骤：

- ① 在各个节点服务器上安装和部署独立服务器模式的 Zabbix 监控系统。
- ② 根据实际需要，安装和部署隶属于各个分布式节点的 Zabbix 服务器代理节点。
- ③ 将各个节点的 Zabbix 服务器由独立服务器模式转换成分布式节点模式。
- ④ 通过各个节点的 Web 前端页面，添加和配置各自的主节点和子节点。

上述的第1、2步，我们已经在前面章节中做过很详细的介绍，在这里就不再复述。因此，在进行第3步操作之前，请读者参阅我们在前面章节中所做的介绍，在各台打算被部署成分布式节点的服务器上安装、部署好独立服务器模式的 Zabbix 系统，并根据需要安装和部署好相应的 Zabbix 服务器代理节点。

接下来，需要将打算部署成分布式节点的独立服务器转换成分布式节点模式。因此，需要在每台需要部署成分布式节点的服务器上执行以下操作：

① 修改 Zabbix 服务器端组件的配置文件 `zabbix_server.conf` 中的 `NodeID` 配置项。该配置项用于指定分布式节点的 ID 号（建议将主节点的 ID 号指定为 1，但是，这不是必需的），有效值为 0~999 的正整数，且在整个分布式监控系统中不得有重复。当该配置项的值为 0 时，则表示对应服务器为独立服务器，否则就为一台分布式节点服务器。

② 关闭服务器端进程。

```
#直接 kill 掉 zabbix_server 进程，可能要重复执行多次。
shell> killall zabbix_server
```

③ 转换数据库，以满足分布式节点的要求。

```
shell> /opt/zabbix/sbin/zabbix_server -n <node_id> -c /opt/zabbix/etc/zabbix_server.conf
```

其中参数 `node_id` 即为分布式节点的 ID 号，也就是我们在前面第1步中 `NodeID` 配置项中指定的值，每个分布式节点所配置的值均不相同。例如，如图 7-5 所示，即为转换主节点数据库时的终端截图。

```
[root@water ~]# /usr/local/sbin/zabbix_server -n 1 -c /usr/local/sbin/zabbix/etc/zabbix_server.conf
..... done.
Converting foreign keys ..... done.
Converting tables ..... done.
Creating foreign keys ..... done.
Conversion completed successfully.
```

图 7-5 分布式节点数据库转换截图

注意：1. 上面第3步所执行的命令，每个节点只能被执行一次，重复执行会出错。在执行之前，建议先备份数据库中的数据。如若数据库转换不成功，则需重新导入数据后重新转换。
2. 在执行上述数据库转换时，为防止有用户访问 Web 前端页面，建议将对应的 Web 服务停止掉。

④ 启动节点上 Zabbix 服务器端进程。

```
shell> /opt/zabbix/sbin/zabbix_server -c /opt/zabbix/etc/zabbix_server.conf
```

当执行节点上 Zabbix 服务器模式转换时，需要记录下每个节点被分配的节点 ID 号。当完成所有节点上 Zabbix 服务器模式的转换后，就可以通过每个节点上的 Web 前端页面，将这些节点添加到 Zabbix 系统中，并配置好它们之间的相互关系。

首先，我们来配置“主节点”。登录到打算将其配置成根主节点的节点 Web 前端页面，并依次选择菜单项“高级配置”→“节点管理”，并在“配置代理服务器”页面的右上部下拉菜单里选择“节点”菜单项。在打开的页面上，可以看到系统已经为我们创建了一个默认的本地节点。单击这个默认节点的节点名称上的超链接，系统将打开“配置节点”页面。通过这个页面，需要确认根主节点的属性配置是否正确，如图 7-6 所示。

图 7-6 所示的配置节点页面上的表单内容比较简单。其中，名称 (Name) 即为节点的名称，在整个多级分布式监控系统中，节点的名称不得有重复；ID，即为节点的 ID 号，需要与配置文件 `zabbix_server.conf` 中 `NodeID` 配置项所指定的节点 ID 号保持一致；类型 (Type)，用于指

定节点的类型，可选的类型有“主节点 (Master)”和“子节点 (Child)”；主节点 (Master node)，如果所配置的节点属于子节点，则通过这个表单项来指定它的主节点；IP 地址，这个表单项用于指定主节点服务器的 IP 地址；端口 (Port)，用于配置主节点的服务器进程所侦听的端口号。

名称

master

ID

1

类型

Local

Master node

Local node

IP地址

127.0.0.1

端口

10051

图 7-6 配置根主节点表单页面截图

确认根主节点的配置无误后，单击“配置节点”页面上的 Create node 按钮，系统就会打开如图 7-6 所示的配置节点的表单页面。通过这个表单页面，我们在根主节点上创建其他的子节点，创建完成后，各子节点的配置信息如图 7-7 所示。

ID	名称	类型	IP地址:端口
1	/master	Local	127.0.0.1:10051
2	/master/Child_1	Child	192.168.5.139:10051
4	/master/Child_2	Child	192.168.6.10:10051
3	/master/Child_1/child_2	Child	192.168.10.158:10051

图 7-7 分布式节点配置信息列表

在根主节点上配置好各子节点的信息后，还需要通过其他节点的 Web 前端页面完成相应节点的配置。当所有节点上的配置都完成了以后，就配置好了一套多级分布式监控系统。

当完成多级分布式监控系统的配置后，可以在根主节点的 Web 前端页面上管理它的子节点的监控元素的配置信息，或查看它的子节点的监控项数据。方法是，通过页面上的“当前节点 (Current node)”下拉菜单，选择需要管理或查看的节点，如图 7-8 所示。

Current node

Child_1

Select Nodes

All

master

Child_1

child_2

查找

图 7-8 选择需要管理的节点

7.4 本章小结

本章介绍了 Zabbix 系统所提供的两种分布式解决方案，即单级分布式解决方案和多级分布式解决方案。多级分布式监控系统解决方案，是在单级分布式监控系统解决方案的基础上发展起来的，其是对单级分布式监控系统解决方案的进一步发展和扩充。因此，它适应于更大范围的系统监控，也可以监控更多的主机。但是，因为在多级分布式解决方案中，主节点服务器数据库中要保存它的所有子节点所采集的监控数据和配置信息数据；所以，当子节点的数量比较多时，根主节点数据库保存的数据量将是非常大的。因此，根主节点对于服务器的硬件要求会相对比较高一些，特别是根主节点的数据库服务器对硬件性能的要求则更高一些。而如果根主节点服务器的硬件性能配置不是很高，则完全可以将根主节点仅作为管理节点来使用，而不用它来具体负责监控被监控主机，虽然，根主节点也完全可以像其他的子节点那样，实施监控数据的采集、分析和计算。

第 8 章 Zabbix 系统优化

Zabbix 系统也和其他许多系统一样，系统的默认配置是针对一般场景所做的。所以，如果 Zabbix 系统配置的监控元素比较多，而 Zabbix 服务器的硬件性能又不是很高，那么针对 Zabbix 系统做一些优化，可能是不得不做的一项重要工作。

笔者始终认为，在对应用系统优化时，应该针对应用系统的特点做有针对性的优化。因此，在本章，首先尝试对 Zabbix 系统的特点做一些分析，并在此基础上提出在对 Zabbix 系统作优化时应该遵循的几个原则。之后，分别从操作系统层面、数据库层面和 Zabbix 系统组件层面，针对 Zabbix 系统的优化展开较深入探讨。

8.1 Zabbix 系统特点分析

系统调优似乎是一个很热的词，几乎所有运维类岗位的招聘要求中都雷打不动地有这一条，要求应聘者会进行系统调优。但是，笔者始终认为，要对一个系统进行很好的调优，就需要对这个系统的特点进行分析，研究它属于哪种类型的系统，然后有针对性地进行系统调优，而不能只是从互联网上找一些调整系统参数的资料，然后按照其上面所述，不知所以然地依葫芦画瓢地进行参数调整。例如，静态网站类系统的调优和重数据库类应用的调优，在调优的针对性和侧重点方面就有很大的不同，而重读取类应用和重写入类应用的调优，在调优的方向和侧重点上也有很大的不同。鉴于此，在具体介绍 Zabbix 系统调优之前，我们先尝试对 Zabbix 系统的特点进行一些分析。

1. 重数据库类型应用

Zabbix 系统与很多网站类应用，特别是静态页面类网站应用有很大的不同，Zabbix 系统是一个严重偏重数据库操作类的应用系统。我们知道，在 Zabbix 系统中，除了要将被监控主机和监控项目等的配置信息保存在数据库中，还要将系统所采集的所有监控数据（极少数历史数据和趋势数据保留天数为零天的监控项目除外）都保留在数据库中，而且数据量的增长和更新非常快。即使“管家（Housekeeper）”进程会定期清理历史数据，从而使数据库中数据的记录条数在达到一定的值后会保持相对稳定，但是，这只是数据的总量基本维持不变，而数据的更新频率仍然是很高的。

也正是因为，在 Zabbix 系统中，所有的被监控主机和监控项目的配置信息，以及系统所采集的所有监控数据都记录在数据库中，所以，如果数据库的性能出现了严重瓶颈，不但会导致监控数据无法及时被写入，从而导致数据图出现大量的断图，而且严重的时候会出现大量警报信息的误报或漏报。因此，Zabbix 系统是一种严重偏重于数据库的应用系统。所以，在对 Zabbix 系统进行调优时，我们更应该将主要精力放在数据库调优方面。同时，如果 Zabbix 服务器端组件和它的数据库分别部署在不同的物理服务器上，则在服务器硬件性能方面，Zabbix 系统数据

库服务器对服务器硬件性能的要求,也要比 Zabbix 系统的其他组件对服务器硬件性能的要求要高。也就是说,如果将 Zabbix 服务器端组件和数据库分开部署,则应该将硬件性能好的服务器用于部署数据库,而不是 Zabbix 服务器端组件。

2. 偏写入型应用

与绝大多数网站类应用不同的是,Zabbix 系统属于偏写入型的应用。虽然,Zabbix 用户也需要经常通过 Zabbix 系统的 Web 前端页面配置被监控主机和监控项目以及查看监控数据等。但是,一般来说,Zabbix 系统是属于组织使用的内部系统,不会有大量用户访问和使用其 Web 前端页面,而仅有组织内部的系统管理或相关人员会通过 Zabbix 系统的 Web 前端页面访问 Zabbix 系统。而与之相对应的是,只要 Zabbix 系统监控了一定数量的被监控主机和监控项目,则它的服务端进程每纳秒都会采集大量的新的监控数据,而这些新采集的监控数据都需要写入到数据库中。

“管家”进程虽然不是用来向数据库中写入数据的,而是根据条件删除数据库中数据的。但是,删除操作和写入操作在读写类型上是相同的,都属于“写”操作。在这一点上,它们与查询属于“读”操作不同。

综上所述,Zabbix 系统是属于偏写入型的应用系统。所以,它的调优侧重点与偏读取型应用系统调优的侧重点应有所不同。同时,也由于 Zabbix 系统的 Web 前端页面的访问量和访问用户数一般来说都比较小,所以,Zabbix 系统的 Web 前端组件对服务器的硬件性能要求不高。因此,如果有必要,完全可以将 Zabbix 系统的 Web 前端组件单独部署在一台虚拟机上,以减小它对 Zabbix 服务器性能的影响。

3. 及时性要求高

作为一款开源的监控系统,Zabbix 系统的一个非常重要的功能就是监控并及时发现被监控主机所存在的故障,同时及时将发现的故障以报警信息的形式通知相关负责人。因此,Zabbix 系统对及时性有比较高的要求是不言而喻的。因而,在对 Zabbix 系统进行调优时,如何提高它的及时性,也应该成为我们调优时不可忽视的方向。

4. 不同监控数据采集方法性能差别大

在前面章节中,我们对于 Zabbix 系统中各种监控数据的采集方法逐一做了介绍和说明。由 Zabbix 系统能提供多达 13 种监控数据采集方法来看,Zabbix 系统监控数据采集功能之强大可见一斑。然而,这些监控数据采集方法的执行效率却差别非常大。比如,通过外脚本程序采集监控数据,就非常占用 Zabbix 服务器的硬件性能。而同样是通过被监控设备代理采集监控数据,使用主动方式在性能方面要比使用被动方式高很多。因此,虽然 Zabbix 系统为我们提供了很多种监控数据采集方法,但是,在实际维护 Zabbix 系统的过程中,我们应该尽量选用性能高的监控数据采集方法。

8.2 Zabbix 系统调优原则

系统调优是一项对知识、经验要求都很高的技术工作,同时,在对一个系统实施调优时,应该结合这个系统的系统特点遵循一定的原则。笔者建议在对 Zabbix 系统进行调优时,应遵循以下调优原则。

1. 尽可能单独部署各个组件

通过前面章节的介绍,我们已经了解了 Zabbix 系统是由多个组件组成的,如 Zabbix 服务器端组件、Web 前端组件、Zabbix 服务器代理组件和被监控设备代理组件等。在安装部署 Zabbix 系统时,这些组件应尽可能分开部署在不同的物理或逻辑服务器上。例如,如前文所述,将 Zabbix 服务器端组件和它所使用的数据库、Web 前端组件分别部署在不同的服务器上。这样,不但能减小这些组件之间的相互影响,而且还可以将性能最好的硬件用在最需要使用的地方,从而最大化地使用服务器硬件性能。比如,将硬件性能最好的服务器用于搭建数据库服务器,而 Web 前端组件则只需安装在一台配置不高的虚拟主机上就可以了。

2. 尽可能采取分布式部署

在安装部署 Zabbix 系统时,尽可能地采用分布式部署,这也是一条 Zabbix 系统优化的原则。当 Zabbix 服务器与被监控主机处在不同的网络环境中,尤其是 Zabbix 服务器需要通过公网采集被监控主机上的监控数据时,更应该采用分布式部署,以减小因为网络波动或其他网络性能原因而带来的影响。

3. 时刻关注数据库服务器的性能

如前文所述,Zabbix 系统是一种重数据库类型应用。因此,它对于数据库服务器性能的要求很高。如果数据库服务器的性能不能满足需要,则不但会导致数据图上产生断图现象,而且严重时会产生报警信息的误报或漏报,以致影响到 Zabbix 系统的稳定运行。因此,在日常维护 Zabbix 系统的过程中,我们应该时刻关注 Zabbix 系统所使用的数据库服务器的性能。而且,对于数据库服务器的性能调化,也是我们对 Zabbix 系统进行调优的重点和难点。

4. 尽可能统一监控数据采集方法

如前文所述,虽然 Zabbix 系统支持多达 13 种之多的监控数据采集方法。但是,在很多时候,我们并不需要将这些种类的监控数据采集方法都使用上。在实际监控系统的管理和维护过程中,应尽可能地使用被监控设备代理组件主动模式采集监控数据,并且关闭未用的监控数据采集方法所开启的服务或进程。比如,如果在整个监控系统中,我们没有任何一个监控项目是通过 JMX 协议采集监控数据的,则就可以将在 Zabbix 服务器上运行的 Java 应用程序网关服务给关闭掉。这样做的好处是,尽量减少不必要的对服务器硬件资源的消耗。因为,即便 Java 应用程序网关不会真正采集监控数据,但是只要它被启动了,它就会或多或少地占用一部分系统资源和硬件资源。

5. 配置合理的数据采集时间间隔

我们知道,在配置监控项目时有一个表单项是用于配置对应监控项目监控数据采集时间间隔的。对于监控数据的采集时间间隔,并不是所有的监控项目都是越小越好,而且对于很多监控项目,如果将其监控数据的采集时间间隔设置得过小完全是没有必要的。比如,对磁盘使用容量数据的采集,对于大多数系统来说,完全没有必要将其采集时间间隔设置成 1min 或 30s 这么小,而完全可以将其设置成 30min 甚至数小时。很显然,不管通过哪种方式采集监控数据,只要 Zabbix 服务器采集一次数据,都会有或多或少额外的系统资源开销。因此,合理地配置监控数据采集的时间间隔,对于 Zabbix 系统的调优意义重大。

类似的,对于网络自动发现功能,如果网络中的设备变动不是非常频繁,则也不需要非常频繁地在同一个网络中执行同一个自动发现规则。在很多时候,同一个网络中的设备通常会保

持相对的稳定,而且即使网络中新添加了设备但没有立即被 Zabbix 系统发现并添加为被监控主机,它也不会影响新设备的使用。同时,以我们实际运维经验来看,通常新添加的设备出现故障的可能性非常低,而且因为是新添加的设备,所以即使该设备出现故障,相关管理人员也会及时发现的。所以,我们完全没有必要将运行网络自动发现规则的时间间隔设置得非常小。通常,我们将运行网络自动发现规则的时间间隔设置为一天或更长时间,是完全可以满足实际监控需要的。

同样,低级自动发现功能也存在类似的情况。通常对于一个稳定的系统,它的被监控对象的数量是相对固定的,不会经常性地变动。所以,我们完全没有必要将低级自动发现规则的执行时间间隔设置得很小。虽然对于同一个被监控主机来说,Zabbix 系统频繁地执行一个低级自动发现规则,并不会每次都会有新的监控项目被添加入系统;但是,对于 Zabbix 服务器来说,它每针对一台被监控主机执行一次低级自动发现规则,就意味着它都要做一次或者更多次的数据采集和数据计算。因此,适当地调低自动发现规则的执行时间间隔,对于 Zabbix 系统的调优也非常有意义。

6. 规划好模板

毫无疑问,Zabbix 系统中的模板为我们日常管理系统提供了非常大的方便;而且,Zabbix 系统中的模板可以相互关联,这就使得 Zabbix 系统中的模板一方面具有很大的灵活性,另一方面,如果模板的规则和管理不善的话,则也可能会带来很大的管理困扰。我们知道,不管是将一个模板关联到另一个模板上,还是将其关联到一台主机上,只要对这个模板上的配置做了任何修改,这些修改就需要被同步到关联它的所有主机或模板上。所以,如果模板嵌套关联的层级过多的话,则毫无疑问一旦对最低层模板的配置做了修改,那么修改关联了它的主机和模板上的配置,将非常耗费系统资源,特别是数据库服务器上的系统资源。

7. 及时处理不被支持的监控项目和触发器

前面我们提到过,如果监控项目的状态为不支持状态,Zabbix 服务器会定时地尝试将其状态转变为“启用”状态,直到该监控项目在下次采集监控数据时被发现仍不被支持。所以,如果 Zabbix 系统中存在大量不被支持的监控项目或触发器,则 Zabbix 系统需要花较多的系统资源处理这些项目。所以,对于这类监控项目,如果确认不再需要它们,那就需要及时地将它们从系统中删除掉;而如果仍然需要这些监控项目,则也需要及时查出它们不被支持的原因,并解决掉导致这些监控项目不被支持的故障,从而避免系统消耗不必要的资源来处理这些实际上没有任何作用的监控项目。

8. 注意函数的使用

我们知道,Zabbix 系统提供了大量的计算函数,这些函数既可以应用到触发器的计算表达式中,也可以应用到监控项目的监控数据的采集中。有了这些函数,我们不仅可以书写出复杂的触发器表达式,而且,还可以实现对所采集的监控数据的再利用、挖掘等。但是,这些函数在执行性能上有很大的差别,如果在使用这些函数时不加以注意,可能会严重地影响到系统的性能。例如,我们知道,在 Zabbix 系统中有 min、max 和 avg 这三个函数,这三个函数用于对指定时间段内系统所采集的监控数据分别求最小值、最大值和平均值,如果指定的时间段很长,则就意味着要查询和计算的数据量非常大。这样,不但对系统中的内存消耗很大,而且对系统的 I/O 性能也提出了很高的要求。而相对于上述的这三个函数,last 和 nodata 函数则只对最新

采集的监控数据进行计算，其所计算的数据量当然也小很多。

综上所述，在书写触发器表达式或创建计算类型的监控项目时，如果需要使用系统提供的函数，则需要注意对可选函数的执行性能进行考量。如果同样可以实现功能和满足需要，则尽可能地选择单值计算的函数，而尽可能地少用或不用类似于 `min`、`max` 和 `avg` 等这类多值计算的函数。

9. 注意监控项目数据类型的选择

我们知道，在 Zabbix 系统中，监控项目的数据类型可以支持数值、字符串、文本甚至日志型数据。了解一些编程基础的朋友大概都了解，对于数值类型的数据（包括无符号的整型和浮点型），其在系统中存储和处理时都是固定长度的。而文本型的数据，包括字符串、文本和日志型的数据，因为其长度是不固定的，所以，很显然系统在处理有固定长度的数值型数据时的性能，要比处理不是固定长度的字符串、文本和日志类型数据的性能要高很多。因此，我们在配置监控项目时就需要注意了，应该尽可能地将监控项目的数据类型配置成数值型的。

8.3 操作系统优化

Zabbix 系统是安装部署在操作系统之上的，所以在对 Zabbix 系统进行调优时先对操作系统进行调优是不可缺少的步骤。通过前面的介绍我们知道，Zabbix 系统可以安装在众多操作系统上，而这些操作系统的调优方法可能又不完全一样。所以，我们没有办法将 Zabbix 系统所支持的操作系统调优方法都介绍一遍。因此，在这里仅以 CentOS 5.x 64 位操作系统为例，具体介绍一下对 Zabbix 系统的操作系统进行调优。

8.3.1 I/O 优化

通过前面的分析我们知道，Zabbix 系统是一种重数据库型的应用系统，所以服务器 I/O 性能的高低对于 Zabbix 系统性能的影响很大。为此在安装操作系统时，就需要有意识地进行必要的 I/O 层面的优化。

1. 选择快速的磁盘和磁盘阵列级别

毫无疑问，磁盘的性能对于整个 Zabbix 系统性能的影响是很大的。所以，如果在成本控制方面许可的话，应尽可能地选择高性能的磁盘。比如，如果能使用固态硬盘（SSD），就不使用非固态硬盘；如果能选择 15K 转的磁盘就不选择 10K 转的磁盘；而 SATA、SAS 和 ISCSI 磁盘的性能要比 IDE 磁盘的性能高很多，因此在选择磁盘接口时 also 需要注意尽可能地选择性能高的接口型磁盘。当然，一般来说，性能越高的磁盘其价格也就越贵，也就意味着组织所投入的成本就会越大。所以，在硬件选择上要结合成本和性能综合考虑。

与选择高性能的磁盘通常会带来成本的大幅度提高不同，调整磁盘阵列的级别通常不会太大地影响到投入的成本。因此，在 I/O 层面对系统进行优化时，根据系统的特点和组织对系统数据安全性的要求，合理地配置磁盘阵列级别对系统的调优就显得很重要。接下来，我们简单地介绍一下常用的各级磁盘阵列的性能差别。

□ RAID 0（又称为条带集或条带卷）。就是将两块或多块磁盘“串联”在一起，形成一

个更大的磁盘，数据是条带化的且均匀分布在所有磁盘上。RAID0 可以提升数据的读写性能，这个应该很容易理解。当系统在磁盘读写数据时，因为数据是均匀地分布在不同的磁盘上的，所以所有磁盘上的数据都可以被并行读写，即各个磁盘的 I/O 可以并行处理。同时，因为 RAID0 是将所有磁盘“串联”在一起的，所以一般情况下，使用 RAID0 所组成的磁盘阵列组的容量就是各个磁盘容量之和。但是，有一种情况特殊，RAID0 是可以允许使用不同容量的磁盘来组建的，但是以不同容量的磁盘组建 RAID0 级别的磁盘阵列时，组建后所形成的磁盘容量是以最小容量的那块盘的容量为准进行计算的。例如，用一块 300GB 硬盘和一块 100GB 的硬盘组成 RAID0 级别的磁盘阵列，则组成的磁盘阵列最后的容量不是 $300+100=400\text{GB}$ ，而是 $100*2=200\text{GB}$ 。这种容量上的限制，在某些以软件实现的磁盘阵列上可能不会出现，例如 Linux 软 RAID 就可以使用全部的空间。

在所有磁盘阵列的级别中，RAID0 的速度是最快的，理论上组成 RAID0 的磁盘越多，速度就越快，它的速度等于所有磁盘的速度之和。但是，实际上还要受到系统 I/O 总线瓶颈的限制。组成 RAID0 的所有磁盘是“串联”起来的，数据是以条带状均匀分布在所有磁盘上的，所以 RAID0 对数据来说是没有冗余的。换句话说，如果组成 RAID0 的磁盘当中有一块坏了，那么所有的数据都将无法恢复了，所以它的数据风险也是最大的。

- ❑ **RAID1（又叫作镜像）。**就是将至少两块磁盘“并联”在一起，组成一个磁盘组。数据被完全镜像地保存在磁阵的每块磁盘上，所以在磁阵中的每块磁盘上都保存了完整的数据备份。因此，除非拥有相同数据的主从盘同时损坏，否则只要有一个磁盘工作正常，即可维持系统的正常运行。所以，RAID1 具有很高的数据可靠性。

因为 RAID1 中的每个磁盘组中的所有磁盘都完整地保存了所有数据，所以系统在读写数据时，主从磁盘都会同时工作，因此 RAID1 的数据读写性能基本上等于单块磁盘的读写性能，在某些情况下会略低一些。

- ❑ **RAID5。**RAID5 是对数据在数据块级别上进行条带化后，再将数据分布地存放在磁盘阵列的所有磁盘上。同时，RAID5 会对存放在各块磁盘上的数据进行奇偶校验，并将奇偶校验数据分布地存放在阵列的所有磁盘上。所以，在 RAID5 级磁盘阵列中任何一块磁盘损坏，系统都可以根据分布地存储在各块磁盘上的奇偶校验数据来恢复损坏磁盘上的数据。同时，RAID5 在读取数据时，正常情况下是无须读写奇偶校验位的数据的。所以，在 RAID5 中，读取数据的性能基本与 RAID0 是差不多的。因而，RAID5 是一种存储性能、数据安全和存储成本兼顾的存储解决方案。

- ❑ **RAID1+0(RAID10)。**这是一种复合的阵列技术，具体来讲，就是先将数据作镜像，也就做成 RAID1，然后再将各镜像组的数据条带化，做成 RAID0。做成 RAID10 以后，如果其中一块硬盘损坏了，则仅有损坏磁盘的那组只有单块盘运行，其他的磁盘则能工作正常。更换损坏的磁盘时，无须停止系统的工作。

- ❑ **RAID50。**与 RAID10 一样，RAID50 也是一种复合的阵列技术，它由条带化的 RAID5 组成。当需要在一个系统中使用很多磁盘时，使用 RAID50 可能是 RAID5 的经济性和 RAID10 的高性能之间的一个折中方案。它的主要用途是存放非常庞大的数据，例如数据仓库等。

综上所述，各级别的磁盘阵列在性能、数据安全方面的比较如表 8-1 所示。

表 8-1 各级别磁盘阵列性能、安全比较表

磁盘阵列级别	概要	冗余	磁盘数	读取速度	写速度
RAID0	成本低、速度快、危险	无	N	快	快
RAID1	速度慢、简单、安全	有	2	慢	慢
RAID5	速度、安全折中（读取速度快，写速度慢）	有	$N+1$	快	慢
RAID10	成本高、快速、安全	有	$2N$	快	快
RAID50	为极大的数据存储服务	有	$2(N+1)$	快	快

因为 Zabbix 系统是属于组织内部使用的非关键性业务系统，通常如果对监控系统的要求不是很高，则允许 Zabbix 系统在一定时间内不可用。因此，结合表 8-1 中所列的各级别磁盘安全性和性能比较的数据，在做好日常备份的情况下，给 Zabbix 系统选择 RAID0 级别的磁盘阵列，可能是最经济性能也最高的解决方案。当然，如果对 Zabbix 系统中历史数据的安全性要求很高，则可能不宜选择 RAID0 级别的磁盘阵列。因为，我们日常在对 Zabbix 系统做数据备份时，通常会忽略历史表和趋势表。这里因为，这两类表是 Zabbix 系统数据库中数据量最大的两类表，对它们进行备份通常会严重影响到系统的性能。在这种情况下，选择 RAID10 级别的磁盘阵列或许是最佳的选择。

2. 调整数据块大小

前面分析过，Zabbix 系统属于一种重数据库型的应用系统，所以对 Zabbix 系统的 I/O 性能优化很重要。下面介绍一下如何从数据块层面对 Zabbix 系统作 I/O 性能优化。

当 Linux 系统保存一个文件到磁盘上时，文件内容会按照一定的大小（大小与文件系统有关，是在文件系统格式化时指定的）被分割成许多个数据块，且整个文件内容是以数据块的形式保存在磁盘上的。同一个数据块在磁盘上的空间分布是连续的，但是用于保存同一个文件内容的多个数据块，在磁盘上的位置可能是不连续的。而我们知道，机械磁盘是通过磁头在盘片上来回滑动（寻址），来定位指定的扇区并读写指定扇区的内容，从而达到读写指定文件内容的目的。这样，对于指定大小的文件，如果分割它的数据块空间越大，则该文件最后被分割成的数据块的数量就越少，从而磁盘在读写文件时磁头在盘片上来回滑动的次数也就越少，且磁头在盘片上来回滑动的幅度也会减小（对于读写某个特定文件来说，磁头来回滑动的幅度可能不会减小，但是对于读写大量的文件来说，磁头滑动幅度总和是一定减小的）。对于机械磁盘来说，如果在其读写文件时减少其磁头在盘片上来回滑动的次数或减小其滑动的幅度，无疑将大大提升其读写数据的速度。而对操作系统来说，如果磁盘的读写速度提升了，也就意味着提升了磁盘 I/O 性能。

在 Linux 操作系统下，当格式化一个文件系统时，系统是允许我们指定该分区的数据块大小的。而在格式化文件系统时所指定的数据块大小，也就是系统在保存文件时该文件内容被分割的数据块的大小。因此，当在格式化一个文件系统时，所指定的数据块的容量越大，则系统在读写该文件系统时的 I/O 性能就越高。

但是，上述所得出的结论实际上是基于一个假设条件的。那就是，文件内容长度大于一个数据块所能存储的最大内容长度，也就是该结论是在一个文件被分割成多个数据块存储的条件

下得出的。而如果所存储的文件很小，一个或者少数几个数据块就可以存储得下，则情况就完全不一样了。为什么会这样呢？其实道理也很简单，我们不难理解，数据块越大，则其在盘片上所分布的空间范围也就越大，相应的，磁头在盘片上移动一个数据块的区域所花的时间也就越长。所以，当磁盘在读写一个文件时，其磁头在定位目标区域时所移动的非目标区域的空间也就越大，相应的，寻址所花费的时间也就越长。因此，当存储小文件时，数据块被设置得越大，I/O 性能也就越低。

那么，你或许会问，多大的文件就为小文件，而大于多大的文件又为大文件呢？关于这个问题，笔者认为没有一个十分严格的规定和标准。一般我们认为，类似于网站和论坛类应用，其对应的文件都是小文件，因为网站上使用的文件根据经验一般只有几 KB 到几十 KB 范围，超过 1MB 大小的文件一般很少；而对于数据库和视频类的应用，其所对应的文件则是大文件，因为这类应用使用的文件少则几 MB 多则几百 GB。

我们在前文分析过，Zabbix 系统是一种重数据库型的应用系统。所以，毫无疑问 Zabbix 系统是属于大文件的系统。但是，不幸的是，Linux 系统在格式化文件系统时，默认设置的数据块大小是其所支持的最小值，即 1024B。所以，为调整系统的 I/O 性能，在格式化文件系统时，需要使用类似于下列命令，指定数据块大小为其所支持的最大值 4096B。

```
shell> mkfs -t ext3 -b 4096 /dev/sdal
```

不过需要注意的是，文件系统的数据块的大小只能在格式化文件系统时指定，一旦文件系统已经格式化了，则就不可再调整数据块的大小了，除非重新格式化文件系统。

如果要查看某个文件系统被配置的数据块大小，则可以执行如下命令：

```
shell> blockdev --getbsz /dev/sdal
```

3. 调整 I/O 调度器

当系统中有进程需要从磁盘中读取数据或者向磁盘上写入数据时，实际上操作系统并不是立即执行相应的读或写入操作的，而是通过一定的算法合并多个 I/O 请求后，再执行具体读或写操作。操作系统的这类合并多个 I/O 请求的算法就称之为 I/O 调度算法，相应的程序就被称之为 I/O 调度器。Linux 2.6.x 内核的操作系统支持以下四种调度器，这些调度器针对不同的应用和场景，具有不同的特性。

- **anticipatory 调度器**。anticipatory 曾经是 Linux 2.6 内核的标准 IO 调度器。Anticipatory 单词的中文含义为“预料的，预想的”，这个词的含义的确揭示了这个算法的特点。简单地说，当系统中有新的 I/O 请求产生时，调度器将产生一个默认的 6 毫秒猜测时间，猜测下一个 IO 请求要干什么，一旦这两个先后的 I/O 请求是相似的，则系统就合并这两个 I/O 操作，以减少磁头来回寻址的时间。所以，该调度器在向磁盘上写数据时会有约 6ms 的等待时间。这对于绝大多数应用有很好的性能，如文件服务器和 Web 服务器，但是对于数据库服务器，该调度器的性能效果却很糟糕。

- **完全公平队列调度器(Complete Fair Queuing, CFQ)**。CFQ 调度器为每个进程/线程单独创建一个队列，用于管理对应进程所产生的 I/O 请求。也就是说，每个进程/线程都拥有一个自己的 I/O 队列，各队列之间的 I/O 使用时间片来调度，以保证每个进程/线程都能公平地被分配到 I/O 操作机会，CFQ 调度器每次执行一个进程的 4 次请求。在 2.6.x 版本的内核中，CFQ 被选择为默认的 I/O 调度器。对于通用的服务器，使用 CFQ 调度器是最好的选择。CFQ 调度器试图均匀地分配对 I/O 带宽的访问，以避免某些进

程被饿死，并实现较低的延迟。CFQ 调度器对于多媒体应用（video、audio）或桌面系统是最好的选择。

- ❑ **截止时间调度器(deadline)**。截止时间调度器，是一个带有截止时间算法的循环调度器。即对于一个 I/O 请求，截止时间调度器确保在对应的截止时间之前服务它。而这个截止时间是可调整的，默认情况下读操作的截止时间时长短于写操作的截止时间时长，这样就防止了写操作因为某些数据不能被读取而饿死的情况发生。截止时间调度器对于数据库系统，例如 Oracle、MySQL 等来说，是最好的选择，同时它也非常适用于应用对实时性要求比较高的场景下，以及大磁盘 I/O 吞吐量的场景下。
- ❑ **NOOP 调度器**。这个调度器只是一个简单的先进先出（FIFO）队列调度器，它不对 I/O 请求做任何排序，而仅仅只是简单地合并相邻的 I/O 请求。NOOP 调度器假设块设备有自己的调度算法，如 SCSI 的 TCQ 等，或者块设备不需要来回移动磁头寻址，比如闪存或 RAM 等。所以，NOOP 调度器也是这类设备的最好选择。

如果要查看当前系统内核正在使用哪种调度器，则只需执行如下命令：

```
shell> cat /sys/block/sda/queue/scheduler
```

执行上述命令后，系统将产生如图 8-1 所示的输出。

```
[root@master ~]# cat /sys/block/sda/queue/scheduler
noop anticipatory deadline [cfq]
```

图 8-1 查看当前系统中正在使用的调度器

由图 8-1 可以看出，当前系统正在使用的调度器为 CFG，即中括号中所显示的调度器。而当前系统所支持的调度器有 NOOP、Anticipatory、Deadline 和 CFG。

修改操作系统使用的 I/O 调度器，没有修改文件系统数据块大小那么麻烦，我们完全可以对正在运行着的系统所使用的 I/O 调度器作出修改，而不需要重启操作系统。只是，使用这种方法对系统所使用 I/O 调度器所作的修改，其效果是临时的，一旦操作系统被重启，则其又会使用默认的 I/O 调度器。当然，我们也可以通过修改启动加载器的配置内容，添加上 I/O 调度器内核参数以达到永久修改系统 I/O 调度器的目的。当前，在 CentOS 发行版中，使用 Grub 加载器应该还是占大部分的。所以，在下面的例子中，以修改 Grub 配置为例，介绍如何对系统 I/O 调度器作永久的修改。

通过上面的介绍我们知道，对于重数据库型应用的 Zabbix 系统，使用默认的 CFQ 调度器对于其来说 I/O 性能并不是最佳的。所以，为了提升系统的 I/O 性能，需要调整系统使用的 I/O 调度器，将其调整为截止时间调度器。而要修改系统的 I/O 调度器可以分临时修改方法和永久修改方法。要临时修改系统使用的 I/O 调度器，只需执行如下命令即可：

```
shell> echo "deadline" >/sys/block/sda/queue/scheduler
```

通过上述方法对操作系统所使用 I/O 调度器进行的修改，在操作系统重启后就会丢失，即操作系统在重启后又将会使用默认的 I/O 调度器。而要永久地修改操作系统所使用的 I/O 调度器，则需要修改操作系统的启动加载器配置文件，这里以修改 Grub 的配置文件为例，介绍一下如何永久性地修改操作系统所使用的 I/O 调度器。永久性地修改操作系统所使用的 I/O 调度器的方法是：编辑/boot/grub/grub.conf 文件，在其中加载内核行添加上内核参数 elevator=deadline 的内容。例如，该文件中内核加载行原来的内容是：kernel /vmlinuz-2.6.18-194.el5 ro root=LABEL=/rhgb quiet，现只需将其修改成 kernel /vmlinuz-2.6.18-194.el5 ro root=LABEL=/elevator=

deadline rhgb quiet，并保存后重启操作系统即可。

8.3.2 Linux 内核参数优化

Linux 系统内核有着大量的可调整的内核参数，这些内核参数从不同方面影响着整个系统的性能。大多数 Linux 发行版这些内核参数的初始配置，都是针对通用应用场景和通用硬件环境的。所以，为了使操作系统内核参数的配置，更匹配我们特定的应用和特定的硬件环境，作为一名系统管理人员，通常需要对 Linux 内核参数进行一些调整和修改，即所谓对 Linux 操作系统内核参数进行优化，以使整个操作系统能够发挥出更好的性能。

Linux 系统内核参数调整方法很简单，只需编辑/etc/sysctl.conf 文件，并修改或添加内核参数配置项及其对应的值，然后执行 sysctl -p 命令即可。以下是我们针对 8 核 2.4G CPU、16GB 内存、双千兆网卡的 Zabbix 服务器所做的内核参数优化列表，可供读者参考。

下面这个内核参数，用于配置 TCP 连接在丢失数据包时的重发机制。当这个参数项被配置成“1”时，TCP 连接的数据接收端会记住在接收数据包时哪些数据包丢失或者接收不正确。这样，数据发送端只会重发那些丢失或传输不正确的数据包给数据接收端。例如，数据发送端给某个数据接收端总共发送了 10 000 个数据包，其中，第 1~3000 和 5000~10 000 个数据包，都被数据接收端正确接收了，而 3001~4999 个数据包丢失或数据接收端经校验后发现所接收到的数据包为不正确的数据包。此时，数据接收端就会通知数据发送端，让它重新发送第 3001~4999 号数据包给自己。同样的场景，如果这个参数项被设置成“0”，则数据接收端会要求数据发送端重发第 3001~10000 号数据包给自己。所以，当启用这个参数项（即将这个参数项的值设置为“1”）时，可以减少系统在网络通信时数据重发的数据量，提高网络利用率。特别是在网络性能不是很好的网络里，启用这个参数项，对网络利用率的提升将更加明显。

```
net.ipv4.tcp_sack = 1
```

下面这个内核参数用于指定，系统是否开启 TCP 接收窗口缩放功能。当该参数被设置为“1”时，表示开启接收窗口缩放功能；而当这个参数被设置为“0”时，则表示关闭接收窗口缩放功能。

```
net.ipv4.tcp_window_scaling=1
```

下面这个参数用于设置，用于接收 TCP 数据的缓冲区大小。该参数有三个数值，第一个值是缓冲区最小值，第二个值是缓冲区的默认值，最后一个值是缓冲区的最大值。如果系统中所配置的监控项目，都是 Zabbix 服务器直接通过被监控设备代理组件采集监控数据的，或你的 Zabbix 系统环境中有很多 Zabbix 服务器代理，则需要相应调大这个参数值。

```
net.ipv4.tcp_rmem = 4096 4194304 8388608
```

下面这个参数用于设置，用于发送 TCP 数据的缓冲区大小。该参数有三个数值，第一个值是缓冲区最小值，第二个值是缓冲区的默认值，最后一个值是缓冲区的最大值。如果系统中所配置的监控项目都是 Zabbix 服务器直接通过被监控设备代理组件采集监控数据的，或者在你的 Zabbix 系统环境中有很多 Zabbix 服务器代理，则需要相应调大这个参数值。

```
net.ipv4.tcp_wmem = 4096 4194304 8388608
```

下面这个参数用于设置，TCP 连接在进入 TIME_WAIT2 状态后，等待对方关闭连接应答的超时时间。如果网络条件比较好，则可以适当将这个参数值设置得小一点，以提高系统的性能。

```
net.ipv4.tcp_fin_timeout = 1
```

下面这个参数用于配置，如果一个 TCP 连接在连续多长时间内处于空闲状态时，系统将向对方发送探测信息。如果网络条件比较好，则可以将这个参数项的值设置得大一些，以提高系

统性能。我们这里将其配置成 2 小时。

```
net.ipv4.tcp_keepalive_time = 7200
```

下面这个参数项用于设置, TCP/UDP 连接可用的本地端口范围。当系统主动发起一个 TCP 或 UDP 连接时, 需要在本地开启一个 TCP 或 UDP 端口。因为 Zabbix 服务器需要频繁地连接数据库和各类组件以及被监控主机, 所以需要将这个范围值调整成最大值。

```
net.ipv4.ip_local_port_range = 1024 65535
```

下面这个参数项用于指定, 是否允许系统将处于 TIME-WAIT 状态下的 TCP 连接应用于新的 TCP 连接。如果将这个参数项的值设置成 1, 则表示系统可以将处于 TIME-WAIT 状态下的 TCP 连接重新应用于新的 TCP 连接。对于 Zabbix 这种大并发的服务器, 开启这个参数选项(即将其值设置成 1)可以提升系统回收处于 TIME-WAIT 状态 TCP 连接的速度, 从而提高系统性能。

```
net.ipv4.tcp_tw_reuse = 1
```

当将下面这个参数设置成 1 时, 则表示开启快速回收处于 TIME_WAIT 状态的 TCP 连接的功能。因而能提升系统的性能。

```
net.ipv4.tcp_tw_recycle = 1
```

下面这个参数项用于配置, 内核在放弃建立 TCP 连接之前, 向被连接端发送 SYN 包的数量。当 TCP 连接的发起端在发起一个新的 TCP 连接时, 会向被连接端发送 SYN 信号。如果在规定的时间内, TCP 连接的发起端没有收到被连接端所发的回应, 则它将重发 SYN 信号。在网络条件较好的情况下, 可以调小这个参数项的值; 反之, 则需要调大其值。

```
net.ipv4.tcp_syn_retries = 1
```

下面这个参数项与上面的参数项类似, 只是它是用于指定系统发送 SYN+ACK 信号的次数的。

```
net.ipv4.tcp_synack_retries = 1
```

每一个连接请求(SYN 报文)都需要排队, 直至本地服务器接收。下面这个参数用于指定, 每个端口 TCP SYN 状态连接队列长度。如果实际处于 SYN 状态的 TCP 连接请求数多于这个参数所设置的值, 则对应的请求就会被丢弃。

```
net.ipv4.tcp_max_syn_backlog = 262144
```

下面这个参数项用于指定, 系统中最多可以允许有多少个孤儿 TCP 连接。所谓孤儿 TCP 连接是指, 套接字未被关联到任何一个用户文件句柄上的连接, 这种连接一般是指连接处于建立过程中或断开过程中的连接。

```
net.ipv4.tcp_max_orphans = 262144
```

下面这个参数用于指定, 系统中可以同时保持 TIME_WAIT 状态的 TCP 连接的最大数。如果实际处于 TIME_WAIT 状态下 TCP 连接的连接数据超过这个参数项所指定的数字时, 则相应的连接就会被立即清除, 并且在系统日志中打印出警告信息。适当减小这个参数的值, 可以让系统及时回收处于 TIME_WAIT 状态的连接, 以提高系统的性能。

```
net.ipv4.tcp_max_tw_buckets = 6000
```

当系统的服务进程在系统中打开一个侦听端口时, 其可以指定用于接收连接的队列长度。而所指定的队列长度的最大值受系统中下面这个内核参数的限制。因此, 对于主动类型监控项目比较多的 Zabbix 系统, 应适当调大下面这个参数的值, 反之, 就可以调小这个内核参数值。

```
net.core.somaxconn = 262144
```

当网络接口接收数据包的速率比内核处理这些包的速率快时, 网络接口所接收的数据包会被临时放入到一个队列中。下面这个参数用于设置, 系统所允许的临时存放到队列中数据包的最大数目。当系统的突发访问量或者访问并发量很高时, 调高这个参数值的大小就很有好处,

可以让系统将来不及处理的数据先暂存在数据队列中。

```
net.core.netdev_max_backlog = 262144
```

下面这个参数用于设置，所有协议的默认写缓冲区大小。但是，对于特定协议的写缓冲区的大小，可以单独设置并且会覆盖掉这个参数项所设置的值。

```
net.core.wmem_default = 8388608
```

下面这个参数用于设置，所有协议的默认读缓冲区大小。但是，对于特定协议的读缓冲区的大小，可以单独设置并且会覆盖掉这里所设置的值。

```
net.core.rmem_default = 8388608
```

下面参数用于设置，所有协议可以使用的最大读缓冲区大小。但是，对于特定的协议可以单独设定其最大读缓冲区大小。然而，针对特定协议所设置的最大读缓冲区大小不可以大于这个参数所设置的数值，例如上面 `net.ipv4.tcp_rmem` 参数中所设置的最大值，不得大于下面这个参数所设置的值。否则，TCP 协议的最大读缓冲区以下面这个参数所设置的数值为最大值。

```
net.core.rmem_max = 16777216
```

下面参数用于设置，所有协议可以使用的最大写缓冲区大小。但是，对于特定的协议可以单独设定其最大写缓冲区大小。然而，针对特定协议所设置的最大写缓冲区大小不可以大于这个参数所设置的数值，例如上面 `net.ipv4.tcp_wmem` 参数中所设置的最大值，不得大于下面这个参数所设置的值。否则，TCP 协议的最大写缓冲区以下面这个参数所设置的数值为最大值。

```
net.core.wmem_max = 16777216
```

下面这个参数用于开启 TCP 数据包时间戳。开启 TCP 数据包时间戳，可以避免序列号的卷绕。一个 1Gb/s 及以上的网络链路，在系统进行数据传输时，肯定会遇到系统以前已使用过的序列号。时间戳能够让内核接收这种“异常”的数据包。

```
net.ipv4.tcp_timestamps = 1
```

下面这个内核参数也包含三个数值，它们用于指定，系统对 TCP 连接占用系统内存的多少所作的反应。该参数的每个值的单位都是内存页，一般一个内存页的大小是 4KB。当系统中所有 TCP 连接所占用的内存页数低于这个参数所指定的第一值时，则表示系统没有内存压力，系统不会考虑释放 TCP 连接所占用的内存；而如果 TCP 连接所占用的内存页大于这个参数所指定的第二个数值时，则表示系统有一定的内存压力，系统将试图稳定 TCP 连接对系统内存的占用；当 TCP 连接所占用的内存大于这个参数所指定的第三个数值时，系统将会丢弃后续报文。

```
net.ipv4.tcp_mem = 524288 1048576 2097152
```

下面这个参数用于设置，在 TCP 连接的三次握手过程中，如果系统未收到对端发送过来的确认包（即 ACK 包，也就是三次握手过程中的第二阶段），尝试发送应答包（也即 SYN+ACK 包）的次数。当网络性能和条件都比较好时，可以将该参数的值设置得小一些；反之，则应该调大该参数值。但是，如果将该参数的值设置得比较大，可能会带来 DoS 攻击风险。

```
net.ipv4.tcp_synack_retries = 1
```

下面这个参数用于设置，系统可以打开的最大文件句柄数。

```
fs.file-max=65536
```

下面这个参数用于指定，从一个进程发送到另一个进程的消息的最大长度，单位是字节。进程间的消息传递是在内核内存中进行的，不会交换到磁盘上，所以如果调大这个参数项的值，则将会增加操作系统占用的内存数量。

```
kernel.msgmax = 65536
```

下面这个参数用于指定，每个消息队列的最大长度，单位是字节。

```
kernel.msgmnb = 65536
```

下面这个参数用于指定，系统可以分配的共享内存段的总数，单位是页。

```
kernel.shmall = 1048576
```


下面这个参数用于指定，内核可以分配的最大共享内存段的大小，单位是字节。

```
kernel.shmmax = 8589934592
```

8.3.3 关闭非必要服务

如果操作系统不是以最小化安装的，则系统在默认情况下会安装并开启很多实际上我们基本上使用不到的服务。很显然，这些我们根本使用不到的服务，在开启的情况下，它必然会占用一定的系统资源。然而，这些服务对系统资源的占用对于我们来说是毫无实际意义的，是对系统资源的不必要浪费。所以，需要将这些开启了但是实际上根本使用不到的服务给关闭掉，以避免它们对系统资源的无谓占用。下面这个 Shell 脚本就是用于关闭系统中无须开启的服务的，你只需将下面这个脚本保存成一个文件，然后执行它即可。

```
1. #!/bin/bash
2. used_service=("cpuspeed" "crond" "sshd" "irqbalance" "microcode_ctl"
"network" "random" "readahead_early" "smartd" "iptables" "sshd" "syslog"
"mysqld" "zabbix_server" "zabbix_agentd" "php-fpm")
3. for i in `ls /etc/rc[3,5].d/S*`
4. do
5.     current_service=$(/bin/echo $i|/bin/cut -c 15-)
6.     FOUND="0"
7.     for s in ${used_service[@]}
8.     Do
9.         if [ "$X$s" == "$X$current_service" ]
10.        Then
11.            FOUND="1"
12.        Fi
13.    Done
14.    if [ "$X$FOUND" == "X0" ]
15.    Then
16.        /sbin/chkconfig ${current_service} off >/dev/null 2>&1
17.        /sbin/service ${current_service} stop
18.    Fi
19. done
20. echo "Turning off Services OK"
```

上述脚本的功能比较简单，它首先定义了一个数组，用于定义哪些服务是需要保留为开启状态的。通过修改这个数组内的元素内容，可以调整需要保留成开启状态的服务列表。接下来，查看/etc/rc3.d和/etc/rc5.d目录下，以S开头的文件，这些文件就是系统用于开启服务的脚本文件的软连接文件，脚本通过它来判断系统已经开启了哪些服务。然后脚本分别执行chkconfig和service命令，用于设置在开机时不自动启动的服务。

8.4 MySQL 数据库优化

通过前文的分析，我们知道 Zabbix 系统是一种重数据库型的应用。所以，数据库服务器性能的高低对于 Zabbix 整个系统性能的高低具有重大的影响。因此，在日常管理和维护 Zabbix 系统的过程中，应该经常关注 Zabbix 系统数据库的性能。

广义上 MySQL 数据库的优化，至少包括数据库 Schema 设计的优化、查询语句的优化、表分区与服务器配置的优化等。然而，Zabbix 是一款由外国组织开发的开源的监控系统，对于其

数据库 Schema 设计的优化和查询语句的优化,显然不是我们关注的方向。因为,除非你已经对 Zabbix 系统的源代码进行了深入的研读,并且有能力对它们进行修改,否则,即使你发现了 Zabbix 系统数据库的 Schema 或者查询语句有需要优化的地方,你也没有办法对它进行修改和调整。因此,这里我们主要简单介绍一下对 Zabbix 系统数据库进行表分区和对服务器配置进行的优化。

8.4.1 MySQL 服务器配置优化

所谓 MySQL 服务器配置的优化,是指通过对 MySQL 服务器参数的调整,以提高 MySQL 数据库性能的过程。这个过程是一个持续的过程,并没有一个终极或者说最佳的配置方案可供我们选择。这是因为,一方面服务器的软硬件配置环境各种各样没有办法做到统一;另一方面,虽然对于 Zabbix 系统来说,其应用的类型是确定的,但是每套系统的数据量、数据类型各不相同;同时,MySQL 数据库可配置的参数非常多,而这些参数值并没有一个理论公式可以计算出来,而往往是根据经验不断调整和优化的。以下是我们在 8 核 2.4G CPU、16GB 内存和硬件 RAID1 的 MySQL 服务器上的配置文件内容,可供读者参考。

```
1. [mysql]
2. #Client Settings
3. port                                = 3306
4. socket                              = /tmp/mysql.sock
5. [mysqld]
6. #Generation Settings
7. user                                = mysql
8. socket                              = /tmp/mysql.sock
9. pid-file                            = /opt/mysql/data/mysql.pid
10. default-storage-engine             = InnoDB
11. max-allowed-packet                 = 16M
12. skip-name-resolve
13. datadir                            = /opt /mysql/data
14. tmp-table-size                     = 32M
15. max-heap-table-size                = 32M
16. query-cache-type                   = 0
17. query-cache-size                   = 0
18. max-connections                    = 500
19. Thread-cache-size                  = 50
20. open-files-limit                   = 65535
21. table-definition-cache              = 4096
22. table-open-cache                   = 2048
23. log-error                           = /opt/mysql/logs/mysql-error.log
24. slow-query-log                      = 1
25. slow-query-log-file                = /opt/mysql/logs/mysql-slow.log
26. #InnoDB Engine settings
27. innodb-data-home-dir                = /opt/mysql/innodb
28. innodb-log-group-home-dir          = /opt/mysql/innodb
29. innodb-additional-mem-pool-size     = 16M
30. innodb-flush-method                 = O_DIRECT
31. innodb-log-files-in-group          = 2
32. innodb-log-file-size               = 256M
33. innodb-flush-log-at-trx-commit     = 2
```



```

34. innodb-file-per-table      = 1
35. innodb-buffer-pool-size   = 12G
36. innodb_buffer_pool_instances = 12

```

虽然, MySQL 数据库可配置的参数有很多, 多达几百个, 但是, 比较幸运的是, 并不是所有的这些参数都需要进行调整和优化, 绝大部分参数我们使用系统默认值就已经很合适了。下面, 我们对于上面配置中部分配置项的含义和作用做一个简单的说明和介绍。

下面这个配置项用于设置数据库的默认存储引擎。虽然 Zabbix 系统是可以支持 MyISAM 存储引擎的, 但是如前面分析的那样, Zabbix 系统是属于偏写入型的应用系统。而 MyISAM 存储引擎只支持数据表级锁。所以, 对 Zabbix 系统来说, 选用 InnoDB 存储引擎获得的性能要比选用 MyISAM 存储引擎高很多。据 Zabbix 官方测试, 对于 Zabbix 系统来说, 其数据库存储引擎选用 InnoDB 时的性能是选用 MyISAM 时的 1.5 倍。

```
default-storage-engine = InnoDB
```

下面这个配置项用于设置 MySQL 服务器可以接收的最大的数据包的大小, 该配置项的默认值是 1MB。因为 Zabbix 系统数据库中存储了图片等内容, 所以, 这里适当增大了这个配置项的值。

```
max-allowed-packet
```

下面这个配置项用于禁止 MySQL 服务器对客户端的 DNS 主机名进行解析。如果配置了这个参数, 则当有客户端连接到 MySQL 数据库时, MySQL 服务器不会尝试通过客户端的 IP 地址反向解析出客户端的主机名, 因此也就不会针对 IP 地址所对应的主机名匹配系统中的授权信息。因为省去了执行反向解析的过程, 所以启用这个配置项, 将会加速客户端连接 MySQL 服务器的速度。况且, 在实际的工作环境中, 可能很少有单位将所有可能连接 MySQL 服务器的客户端都做了反解析, 在这个时候, 如果 MySQL 服务器尝试通过 IP 地址反解析客户端的主机名, 将会使 MySQL 服务器花大量的时间等待 DNS 服务器的响应。所以, 启用这个配置项将会提高 MySQL 服务器的性能, 特别是在大并发的环境下, 所提高的性能效果将更加明显。

```
skip-name-resolve
```

下面这个配置项用于设置内存中临时表的最大大小。当在查询语句中使用了 `group by`、`order by` 等数据聚合语句时, MySQL 数据库会自动在内存中创建临时表, 以用于暂存查询时产生的临时数据。当系统创建的临时表的大小小于 `tmp-table-size` 配置项所指定的大小 (如果配置项 `max-heap-table-size` 所指定的数值小于 `tmp-table-size` 配置项所指定的值, 则临时表的最大大小小于 `max-heap-table-size` 配置项所指定的数值) 时, 则临时表被创建在内存上, 否则临时表被创建在磁盘上。很显然, 内存的访问速度要比磁盘的访问速度快很多, 所以, 在系统内存充足的情况下, 调大这个配置项的值, 可以让系统将更多的临时表创建在内存中, 从而提高系统的性能。

```
tmp-table-size
```

下面这个配置项用于指定最大内存表的大小。所谓内存表是指存储引擎是 `memory` 的表。按说我们在优化 Zabbix 系统数据库时不需要调整这个配置项的值, 因为 Zabbix 系统不会创建存储引擎为 `memory` 的表。但是, 因为 MySQL 数据库在内存中创建临时表时, 其所能在内存中创建的最大的临时表的大小, 是由参数 `tmp-table-size` 和 `max-heap-table-size` 中较小的一个参数值所决定的。所以, 为了能让 MySQL 数据库在内存中创建 `tmp-table-size` 配置项所指定大小的临时表, `max-heap-table-size` 配置项所设定的数值不得小于 `tmp-table-size` 配置项所指定的数值。在这里, 我们将这两个配置项的值设置成一样, 都是 32MB。

```
max-heap-table-size
```

下面这个配置项用于设置查询缓存类型。该配置项的值可以设置为 0, 表示关闭所有的查

询缓存，即所有的查询都不缓存；当设置成 1 时，表示开启查询缓存，除了以 `select sql_no_cache` 开头的查询以外，其他的查询系统都将缓存其所查询到的结果；最后一个可以设置的值是 2，表示仅缓存以 `select sql_cache` 开头的查询语句的查询结果。通过前面对 Zabbix 系统特点的分析我们知道，Zabbix 系统是一种对及时性要求很高的系统，而且我们很容易理解，Zabbix 系统中的数据是实时变化的，所以 MySQL 数据库的缓存功能对 Zabbix 系统来说作用不大。因此，这里将这个配置项的值设置成 0，即关闭数据库的缓存功能，以免因为开启它而造成对系统内存不必要的占用。

query-cache-type

下面这个配置项用于指定，用于缓存查询结果的内存空间大小。因为我们在前面都已经关闭了 MySQL 数据库的查询缓存功能，所以再指定缓存空间大小已经没有任何意义了，因此，这个配置项的值也设置成 0。

query-cache-size

下面这个配置项用于设置 MySQL 服务器可以允许连接的最大并发数，当这个配置项的值设置得过小时，客户端尝试连接 MySQL 数据库时会报 “Too many connections” 错误。

max-connections

下面这个配置项用于设置线程缓存大小。如果这个配置项的值被设置为大于 0 的正整数，则当有客户的连接被关闭时，与该连接相关的线程可以被缓存起来，以备将来新的连接使用。而如果这个配置项的值被设置为 0，或者被缓存的线程数已经达到了配置项 `thread-cache-size` 所设置的值时，则当关闭一个连接时，与该连接相关的线程就会被销毁而不会被缓存。

thread-cache-size

下面这个配置项用于设置 `mysqld` 可以同时打开系统文件的句柄数，也就是系统参数中文件的句柄数。

open-files-limit

下面这个配置项用于指定，在表定义缓存区中可以缓存多少个表定义。当 MySQL 要操作一个表中的数据时，它首先需要打开对应表，并读取该表的定义。这个配置项用于指定最多有多少个表定义可以缓存在内存中。

table-definition-cache

下面这个配置项用于指定 MySQL 所有线程可以同时打开的表的个数，即也就是 MySQL 可以允许的，处于打开状态的表的个数。需要注意的是，因为同一个表可以被不同的线程同时打开，所以，系统中同时处于打开状态的表的个数要远远大于数据库中所定义的表的个数。

table-open-cache

下面这个配置项用于开启记录慢查询功能。

slow-query-log

下面这个配置项用于指定 `innodb` 存储引擎，用于存储数据字典和其他内部结构信息的内存池的大小。对于 Zabbix 系统数据库而言，这个配置项的值是相对固定的。如果这个配置项的值被设置得偏小，则 MySQL 会产生相关的错误信息。

Innodb-additional-mem-pool-size

下面这个配置项用于指定 MySQL 刷新 `innodb` 数据和日志数据到磁盘上的方法。对于 Linux 系统而言，这个配置项的值如果设置为 `O_DIRECT`，则可以让 MySQL 向磁盘上刷新数据时通知操作系统不要缓存数据，而直接由 MySQL 向磁盘上读写相关数据。因此，在 Linux 系统上，将该配置项设置为 `O_DIRECT`，可以提高数据库的 I/O 性能，从而大大提高整个系统的性能。

innodb-flush-method

下面这个配置项用于指定，`innodb` 重做日志（redo log）组中重做日志文件的个数。重做日

志文件中记录的是, MySQL 系统中所执行的尝试修改数据库数据的查询语句。这些语句是以二进制的形式存储在重做日志文件中的。当因为某种原因导致数据库中的数据不正确时, MySQL 会在重启时自动使用这些重做日志文件中所记录的语句, 纠正不正确的数据。这个配置项用于指定系统中保留多少个重做日志文件, 超过这个配置项所指定数目的重做日志文件, 将会被循环覆盖重写。

`innodb-log-files-in-group`

下面这个配置项用于指定, 每个重做日志文件的大小。当单个重做日志文件内所记录的内容大小达到或超过这个配置项所指定的大小时, 系统将切换重做日志文件。调大该配置项的值, 将有利于提高 MySQL 数据库的 I/O 性能, 但是, 随之带来的是当数据库中的数据不完整时, 系统恢复和纠正数据的时间也将加长。

`innodb-log-file-size`

下面这个配置项用于控制 innodb 重做日志写行为。当该配置项的默认值是 1 时, 表示每次执行事务提交操作时系统都将信息写入相应的重做日志, 并且将重做日志的信息及时刷新到磁盘上对应重做日志文件中; 当该配置项被设置为 0 时, 则表示不管系统中是否有事务提交, 系统都会每 ns 写一次重做日志, 并且将重做日志信息刷新到磁盘上相应的重做日志文件中; 而当这个配置项取值为 2 时, 则系统会在每个事务提交时写入重做日志, 但是日志信息每隔一 ns 被刷新到磁盘上相应的重做日志文件中。很显然, 当这个配置项的值被设置成 1 时数据最安全, 因为只要有事务提交, 系统就会将相应的信息写入相应的重做日志中, 并及时地将这些重做日志信息刷新到磁盘上对应的重做日志文件中。但是, 随之带来的是对磁盘 I/O 的占用, 从而导致 I/O 性能较低。与之相对应的是, 当这个配置项的值被设置为 0 和 2 时, 因为系统并不是针对每个事务都会将相应的重做日志信息刷新到磁盘上, 所以会提高系统的 I/O 性能。但是, 因为系统并不是在每个事务提交时, 就及时地将重做日志信息刷新到磁盘上, 所以当这个配置项的值被设置为 0 或 2 时, 在极端的情况下会导致 1~2ns 内的数据丢失。

如前文所分析的那样, 一般来说 Zabbix 系统并不是一个组织内的关键系统, 除非有特殊需要, 否则即使系统有一到两 ns 的数据丢失也是可以接受的。因此, 从提高系统性能的角度出发, 建议这个配置项的值可以设置为 0 或 2。这里将其设置为 2。

`innodb-flush-log-at-trx-commit`

下面这个配置项用于设置, 对于 innodb 引擎的表是否按每个表单独存储数据和索引。如果这个配置项被设置成 0, 则所有存储引擎为 innodb 的表, 其所存储的数据和索引都存储在一个文件中。在这种情况下, 当数据库的表和数据都比较多时, 这个用于存储数据和索引的文件的体积就非常大, 因此不利于系统性能的提升。而当这个配置项的值被设置为 1 时, 则每个 innodb 存储引擎类型的表中所存储的数据和索引信息, 将会单独分开成不同的系统文件进行存储。毫无疑问, 这么做可以减小每个数据文件的大小, 从而提高系统的 I/O 性能。在 MySQL 5.6.6 以前的版本中, 这个配置项的默认值为 0; 而在 MySQL 5.6.6 及其以后的版本中, 这个配置项的默认值为 1。

`innodb-file-per-table`

下面这个配置项用于设置 Innodb 缓冲池的大小。Innodb 缓冲池是系统内存中的一块空间, 它用于在内存中缓存表数据和索引信息。很显然, 尽可能大地设置这个配置项的值, 可以让 MySQL 在系统内存中缓存更多的数据, 从而减少执行 I/O 操作的次数, 因此能够提高系统的性能。如果 MySQL 服务器是专门运行 MySQL 服务的, 没有运行其他服务, 则一般建议这个配置项的值设置为服务器物理内存的 80% 左右。

`innodb-buffer-pool-size`

下面这个配置项用于指定 Innodb 缓冲池被分割成子缓冲池的个数。当 Innodb 总缓冲池的大小，也就是 innodb-buffer-pool-size 配置项所指定的值大于 1GB 时，就应该将这个配置项的值设置为大于 1 的整数。这样做的好处是，当缓冲池被分割成多个独立的子缓冲池时，可以减少不同线程同时读写缓存页而产生的竞争，从而提高系统的并发数。

```
innodb_buffer_pool_instances
```

如前文所述，系统调优是一个系统化和持续化的过程，所以，在这里给出的配置可能并不完全适合你的系统环境，你可能需要根据自己系统的实际软硬件环境，在该配置的基础上进行进一步的调整和优化。

8.4.2 数据库表分区

我们知道，当数据库中某个表的数据量很大时，对该表进行分区就可以提升数据库的性能。对于表分区为何能提升数据库的性能其原理其实也很简单：表分区可以将原先存放在一个系统文件中的数据分割成多个系统文件进行存储，从而减小每个存储文件的大小，以达到提升系统 I/O 性能，增加系统处理并发的能力。

MySQL 数据表既可以支持水平分区，也可以支持垂直分区。然而，如果要对 Zabbix 系统数据库的表进行垂直分区，就意味着需要修改 Zabbix 系统的源代码，而这已经超出了本书介绍的范围。所以，我们这里介绍的分区是针对 Zabbix 系统数据库中某些表所进行的水平分区。

在实际工作中我们发现，对于 Zabbix 系统来说，其频繁地使用的数据一般也就是系统最近采集的监控数据，对于历史时间较长的数据（比如超过几天的历史数据），一般很少使用到，通常也就是用户在查看监控项目的历史数据图时会使用到这部分数据。所以，对 Zabbix 系统来说，对其数据库表进行水平分区，以使其历史数据和最近采集的数据分开存放在不同的系统文件中，可以获得比一般应用系统更大的性能提升空间。

而且，通过前面章节的介绍我们知道，在 Zabbix 系统中有一种叫做“管家”的进程，它是用于维护系统中历史数据和趋势数据的。即，“管家”进程会定期根据配置，删除系统数据中“过期”的历史数据和趋势数据。然而我们知道，通过应用删除数据库中大量的数据时会非常占用系统性能，特别是系统的 I/O 性能。如果对 Zabbix 系统的数据库表进行分区，则就可以通过定期删除“过期”的分区表，来达到删除“过期”历史数据和趋势数据的目的。这样，就可以做到连“管家”进程都不用启动了，从而能够大大提升系统的性能。

一般来说在 Zabbix 系统中，与配置相关的表（比如主机表、监控项目表等）的数据量不会非常大，所以对于这类表我们不需要对它们进行分区。而需要做表分区的表，不同表的数据量也不尽相同，所以，我们对它们进行分区的“粒度”（指表被分区的时间间隔）一般来说也不尽相同。当然，对数据表进行分区的“粒度”也跟系统中数据量的大小有关。如果系统的数据量不大，则分区的“粒度”可以粗一点，否则就要调小表分区的“粒度”。表 8-2 中列出了我们建议应进行分区的表以及它们被分区的“粒度”。

表 8-2 表分区“粒度”列表

序号	表名	分区的粒度	表的作用
1	history	每日	该表用于存储“数值（浮点数）”型监控项目所采集的历史数据

续表

序号	表名	分区的粒度	表的作用
2	history_log	每日	该表用于存储“日志”型监控项目所采集的历史数据
3	history_str	每日	该表用于存储“字符串”型监控项目所采集的历史数据
4	history_text	每日	该表用于存储“文本”型监控项目所采集的历史数据
5	history_uint	每日	该表用于存储“数值（无符号整型）”型监控项目所采集的历史数据
6	acknowledges	每月	该表用于记录事件确认信息
7	alerts	每月	该表用于记录系统发送的报警信息数据
8	auditlog	每月	该表记录审计信息
9	events	每月	该表用于记录系统中所发生的事件信息
10	service_alarms	每月	该表用于记录“IT服务”警告状态
11	trends	每日	该表用于存储“数值（浮点数）”型监控项目趋势数据
12	trends_uint	每日	该表用于存储“数值（无符号整型）”型监控项目趋势数据

表 8-2 列出了 Zabbix 系统数据库需要被分区的表，以及这些表被分区的粒度。很显然，我们这里所建议的各表分区粒度是根据通常情况所作的建议，可能并不完全适用于你的系统。例如，如果你的 Zabbix 系统中“数值（浮点数）”型监控项目并不是很多，或者这类监控项目的监控数据不是很多，这个时候就没有必要对 history 表按每日进行分区。同样，如果你的系统中没有配置“IT 服务”，或者只配置了很少量的“IT 服务”，这个时候 service_alarms 表中的数据量就可能很少，因此也就没有必要对 service_alarms 进行分区或者调大分区的粒度。

确认了需要分区的表以及各个表的分区粒度后，接下来，就需要手工对这些表进行分区。我们知道，要对一张 MySQL 数据库表进行水平分区，就需要选定分区所依据的字段，即对选定字段中存储的内容按照数值范围划分不同的分区，MySQL 依据所划分的范围将被分区的表中内容分成不同的数据文件进行存储。对 Zabbix 系统数据库来说，当对其表进行分区时，选定具有时间属性的字段作为分区的依据将是最好的选择。因为，这样一来方便我们对分区进行管理；二来，可以通过删除超过一定时间长度的历史分区，来达到清除系统中历史数据的目的，从而也就可以停用 Zabbix 系统中“管家”进程了。非常幸运的是，我们所需要分区的表中都包含“clock”字段。因此，在对各表进行分区时，“clock”字段就成为对表进行分区的不二选择。

在具体创建分区之前，还需要做一些准备工作。在 Zabbix 系统数据库中，某些表使用了另外一些表的主键作为其外键。这样，如果在创建表分区之前不消取这些外键的约束，则将无法创建表分区。所以，在创建表分区之前，需要对这些表进行必要的处理。以下语句即是用于处理所要分区表的外键约束的。

```
mysql> alter table 'history_log' drop primary key, add primary key('itemid', 'id', 'clock');
mysql> alter table 'history_log' drop key 'history_log_2';
mysql> alter table 'history_log' drop key 'history_log_1';
mysql> alter table 'history_str' drop key 'history_str_1';
mysql> alter table 'history_text' drop primary key, add primary key('itemid', 'id', 'clock');
mysql> alter table history_text drop key history_text_1;
mysql> alter table 'history_text' drop key 'history_text_2';
mysql> alter table 'acknowledges' drop primary key, add key 'acknowledgedid' ('acknowledgeid');
```

```
mysql> alter table 'alerts' drop primary key ,add key 'alertid' ('alertid');
mysql> alter table 'auditlog' drop primary key, add key 'auditid' ('auditid');
mysql> alter table 'events' drop primary key, add key 'eventid' ('eventid');
mysql> alter table 'service_alarms' drop primary key,add key 'servicealarmid'
('servicealarmid');
mysql> alter table service_alarms drop foreign key c_service_alarms_1;
```

执行上述语句后，再执行下列 SQL 查询就可以对表 8-2 中所列的各数据库表进行手工分区了。

为按日分区的各表创建分区的语句如下所示：

```
mysql> ALTER TABLE <tablename> PARTITION BY RANGE(clock) (PARTITION p20140516
VALUES LESS THAN (UNIX_TIMESTAMP("2014-05-17 00:00:00")),PARTITION p20140517
VALUES LESS THAN (UNIX_TIMESTAMP("2014-05-18 00:00:00")),PARTITION p20140518
VALUES LESS THAN (UNIX_TIMESTAMP("2014-05-19 00:00:00")),PARTITION p20140519
VALUES LESS THAN (UNIX_TIMESTAMP("2014-05-20 00:00:00")));
```

将上述语句中的<tablename>替换成所需要分区的表的名称，并执行上述语句就可以对表进行分区。其中，“p20140519”为分区的名称，而 UNIX_TIMESTAMP 函数的参数则是被分区的时间点。类似的，为按月分区的各表创建分区的语句如下所示：

```
mysql> ALTER TABLE <tablename> PARTITION BY RANGE(clock) (PARTITION p201404
VALUES LESS THAN (UNIX_TIMESTAMP("2014-05-01 00:00:00")),PARTITION p201405
VALUES LESS THAN (UNIX_TIMESTAMP("2014-06-01 00:00:00")));
```

8.4.3 创建自动维护分区存储过程

前面手工创建了各表的分区，以后可以通过执行下面的 SQL 语句，为指定的数据库表添加新的分区或删除不需要的分区。

1. 添加新分区的 SQL 语句

```
mysql> ALTER TABLE 'history_uint' ADD PARTITION ( PARTITION p20140520 VALUES LESS
THAN (UNIX_TIMESTAMP("2014-05-21 00:00:00")));
```

2. 删除历史分区的 SQL 语句

```
mysql> ALTER TABLE 'history_uint' DROP PARTITION p20140516;
```

如果每天能登录到 Zabbix 系统数据库服务器上，则执行上述 SQL 语句来维护系统中表分区信息，当然是没有问题的。只是每天这样手工执行相同的 SQL 语句，不但工作量大、效率低，而且还很容易出错。因此，如果能够编写一个存储过程或脚本，放在系统的定时任务中，由系统每天定时执行，这样不但能够大大提高工作效率，而且还可以避免因为手工操作而带来的错误。以下就是能满足上述这种需求的存储过程代码：

```
1. DELIMITER $$
2. CREATE PROCEDURE 'create_partition' (SCHEMANAME varchar(64), TABLENAME varchar(64),
PARTITIONNAME varchar(64), CLOCK int)
3. BEGIN
4.     DECLARE RETROWS int;
5.     SELECT COUNT(1) INTO RETROWS FROM information_schema.partitions
WHERE table_schema = SCHEMANAME AND table_name = TABLENAME AND partition_name
= PARTITIONNAME;
6.     IF RETROWS = 0 THEN
7.         SET @sql = CONCAT( 'ALTER TABLE ', SCHEMANAME, '.', TABLENAME,
8.         ' ADD PARTITION (PARTITION ', PARTITIONNAME, ' VALUES LESS THAN
(' , CLOCK, '));' );
9.         PREPARE STMT FROM @sql;
10.        EXECUTE STMT;
```



```

11.      DEALLOCATE PREPARE STMT;
12.      END IF;
13. END $$
14. DELIMITER ;

```

上面这个存储过程带4个输入参数，分别是数据库名称 SCHEMANAME，它可以是一个不超过64个字符的字符串；被分区的表名 TABLENAME，同样可以是一个不超过64个字符的字符串；分区名 PARTITIONNAME，可以是一个不超过64个字符的字符串；以及分区范围的截止时间 CLOCK，它是一个正整数的时间戳。上述这段存储过程的源代码不难理解：首先，定义了一个变量 RETROWS，然后查找 information_schema.partitions 表中是否存在属于数据库 SCHEMANAME，分区名为 PARTITIONNAME 的 TABLENAME 分区信息，并将所查找到的个数赋给变量 RETROWS。如果在系统中没有找到将要创建的分区信息，则存储过程执行相应的创建分区的 SQL 语句，否则存储过程返回。

很显然，上面的存储过程是负责具体创建表分区的，它需要被其他存储过程或脚本调用才会在系统中对指定的表创建指定的分区。所以，接下来，需要编写另一个存储过程，用它来调用上面的存储过程，从而在系统中创建指定的分区。下面即为该存储过程的源代码：

```

1. DELIMITER $$
2. CREATE PROCEDURE 'create_next_partitions'(SCHEMANAME varchar(64), TABLENAME
varchar(64), PTYPE int(1))
3. BEGIN
4.   DECLARE NEXTCLOCK timestamp;
5.   DECLARE PARTITIONNAME varchar(16);
6.   DECLARE CLOCK int;
7.   SET @totaldays = 7;
8.   SET @i = 1;
9.   createloop: LOOP
10.    IF PTYPE = 0 THEN
11.      SET NEXTCLOCK = DATE_ADD(NOW(), INTERVAL @i DAY);
12.      SET PARTITIONNAME = DATE_FORMAT( NEXTCLOCK, 'p%Y%m%d' );
13.      SET CLOCK = UNIX_TIMESTAMP( DATE_FORMAT( DATE_ADD( NEXTCLOCK , INTERVAL 1
DAY), '%Y-%m-%d 00:00:00' ));
14.    ELSE
15.      SET NEXTCLOCK = DATE_ADD(NOW(), INTERVAL @i MONTH);
16.      SET PARTITIONNAME = DATE_FORMAT( NEXTCLOCK, 'p%Y%m' );
17.      SET CLOCK = UNIX_TIMESTAMP( DATE_FORMAT( DATE_ADD( NEXTCLOCK , INTERVAL 1
MONTH), '%Y-%m-01 00:00:00' ));
18.    END IF;
19. CALL create_partition( SCHEMANAME, TABLENAME, PARTITIONNAME, CLOCK );
20. SET @i=@i+1;
21. IF @i > @totaldays THEN
22.   LEAVE createloop;
23. END IF;
24.   END LOOP;
25. END $$
26. DELIMITER ;

```

类似的，上面定义的存储过程 create_next_partitions 带3个输入参数，前两个参数分别是不超过64个字符长度的数据库名和表名，第三个参数则是一位整型的参数，它用于指定分区的类型，即是按月分区还是按日分区。当该参数值为0时，表示按日分区；而当该参数值非0时，则表示按月分区。在上述存储过程中执行了一个循环，该循环的次数由变量 totaldays 的值来

指定。例如，上述所给出的代码中 `totaldays` 变量的值为 7，则这个存储过程会执行 7 次循环体。每执行一次循环体，存储过程首先确定出相关变量的值，然后调用 `zabbix.create_partition` 存储过程，在系统中创建指定的分区。

上面是系统自动创建分区时需要使用的两个存储过程，接下来，还需要创建两个存储过程，以用于删除过期的表分区。这两个存储过程的源代码如下：

```
1. DELIMITER $$
2. CREATE PROCEDURE 'drop_partition'(SCHEMANAME varchar(64), TABLENAME
  varchar(64), PARTITIONNAME varchar(64))
3. BEGIN
4. DECLARE RETROWS int;
5. SELECT COUNT(1) INTO RETROWS FROM information_schema.partitions WHERE
  table_schema = SCHEMANAME AND table_name = TABLENAME AND partition_name =
  PARTITIONNAME;
6. IF RETROWS = 1 THEN
7. SET @sql = CONCAT( 'ALTER TABLE ', SCHEMANAME, '.', TABLENAME, ' DROP
  PARTITION ', PARTITIONNAME, ';' );
8. PREPARE STMT FROM @sql;
9. EXECUTE STMT;
10. DEALLOCATE PREPARE STMT;
11. END IF;
12. END $$
13. DELIMITER ;
```

与上面创建分区的存储过程 `create_partition` 类似，删除表分区的存储过程 `drop_partition` 首先也是查看 `information_schema.partitions` 表中是否存在指定的表分区信息。如果存在指定的表分区，则执行相应的删除表分区语句，从而将指定的表分区从系统中删除；如果不存在指定的表分区，则该存储过程不做任何事情直接返回。同样，该存储过程也需要其他存储过程来调用它。调用它的存储过程为 `drop_old_partitions`，其代码如下所示：

```
1. DELIMITER $$
2. CREATE PROCEDURE 'drop_old_partitions'(SCHEMANAME varchar(64), TABLENAME
  varchar(64), PTYPE int(1))
3. BEGIN
4. DECLARE OLDCLOCK timestamp;
5. DECLARE PARTITIONNAME varchar(16);
6. DECLARE CLOCK int;
7. SET @mindays = 7;
8. SET @maxdays = @mindays+7;
9. SET @i = @maxdays;
10. droploop: LOOP
11. IF PTYPE = 0 THEN
12. SET OLDCLOCK = DATE_SUB(NOW(), INTERVAL @i DAY);
13. SET PARTITIONNAME = DATE_FORMAT( OLDCLOCK, 'p%Y%m%d' );
14. CALL drop_partition( SCHEMANAME, TABLENAME, PARTITIONNAME );
15. ELSE
16. SET OLDCLOCK = DATE_SUB(NOW(), INTERVAL @i MONTH);
17. SET PARTITIONNAME = DATE_FORMAT( OLDCLOCK, 'p%Y%m' );
18. CALL drop_partition( SCHEMANAME, TABLENAME, PARTITIONNAME );
19. END IF;
20. SET @i=@i-1;
21. IF @i <= @mindays THEN
22. LEAVE droploop;
```

```

23. END IF;
24. END LOOP;
25. END $$
26. DELIMITER ;

```

上述存储过程源代码中的 `mindays` 变量，用于设置系统中保留多年的历史数据。需要注意的是，当启用上述这个存储过程来管理 Zabbix 系统数据库分区时，则我们在配置监控项目时所指定的“保留历史数据（天）”和“保留趋势数据（天）”配置项就不起作用了。虽然，我们在配置监控项目时仍然可以给这两个配置项配置相应的数值，但是，因为我们不再使用“管家”进程来管理系统中的历史数据和趋势数据，所以，系统中能保留多长时间的历史数据和趋势数据将由上述存储过程中的代码所决定。

稍微学习过编程的读者不难看出，上述这个存储过程中也有一个循环体，该循环体的循环次数由 `maxdays` 变量和 `mindays` 变量决定。而该循环体每循环一次，就执行一次 `drop_partition` 存储过程，具体实施删除指定分区操作。

上面既定义了添加分区的存储过程，又定义了删除分区的存储过程。接下来，还需要创建一个存储过程，用于调用上述添加分区的存储过程和删除分区的存储过程，并代入实际要管理分区的表名和数据库名，其源代码如下：

```

1. DELIMITER $$
2. CREATE PROCEDURE 'create_zabbix_partitions'()
3. BEGIN
4. CALL zabbix.create_next_partitions("zabbix","history",0);
5. CALL zabbix.drop_old_partitions("zabbix","history",0);
6. CALL zabbix.create_next_partitions("zabbix","history_log",0);
7. CALL zabbix.drop_old_partitions("zabbix","history_log",0);
8. CALL zabbix.create_next_partitions("zabbix","history_str",0);
9. CALL zabbix.drop_old_partitions("zabbix","history_str",0);
10. CALL zabbix.create_next_partitions("zabbix","history_text",0);
11. CALL zabbix.drop_old_partitions("zabbix","history_text",0);
12. CALL zabbix.create_next_partitions("zabbix","history_uint",0);
13. CALL zabbix.drop_old_partitions("zabbix","history_uint",0);
14. CALL zabbix.create_next_partitions("zabbix","acknowledges",1);
15. CALL zabbix.drop_old_partitions("zabbix","acknowledges",1);
16. CALL zabbix.create_next_partitions("zabbix","alerts",1);
17. CALL zabbix.drop_old_partitions("zabbix","alerts",1);
18. CALL zabbix.create_next_partitions("zabbix","auditlog",1);
19. CALL zabbix.drop_old_partitions("zabbix","auditlog",1);
20. CALL zabbix.create_next_partitions("zabbix","events",1);
21. CALL zabbix.drop_old_partitions("zabbix","events",1);
22. CALL zabbix.create_next_partitions("zabbix","service_alarms",1);
23. CALL zabbix.drop_old_partitions("zabbix","service_alarms",1);
24. CALL zabbix.create_next_partitions("zabbix","trends",0);
25. CALL zabbix.drop_old_partitions("zabbix","trends",0);
26. CALL zabbix.create_next_partitions("zabbix","trends_uint",0);
27. CALL zabbix.drop_old_partitions("zabbix","trends_uint",0);
28. END $$
29. DELIMITER ;

```

这里定义的 `create_zabbix_partitions` 存储过程其功能比较简单，它主要用来调用在前面定义的 `zabbix.create_next_partitions` 和 `zabbix.drop_old_partitions` 存储过程，以创建指定的表分区和删

除过期的表分区。在这个存储过程中调用的 `zabbix.drop_old_partitions` 和 `zabbix.drop_old_partitions` 函数都有 3 个输入参数，从左到右依次是数据库名称、被分区表的名称以及分区的类型。当第三个输入参数取 0 时，则表示按每日进行分区，分区名为“p 年月日”的形式；而当这个输入参数取 1 时，则表示按月进行分区，并且分区名为“p 年月”的形式。

当创建好上述 5 个存储过程后，在系统的 `crontab` 定时任务中添加上如图 8-2 所示的定时任务，就可以实现每天自动创建新的分区，并删除过期的分区功能了。

```
0 1 * * * /bin/echo "use zabbix;call create_zabbix_partitions()" |/opt/mysql/bin/mysql -uzabbix -pzabbix
```

图 8-2 调用存储过程定时任务信息截图

要使用上述存储过程来管理系统中的表分区，需要注意的一点是，首先要对每个需要分区的表，按照前面所述的方法手工创建相应的表分区，而不能在没有对相应的数据库表创建表分区时，就直接执行上述存储过程。如果这样，系统会报如图 8-3 所示的错误信息。

```
ERROR 1505 (HY000): Partition management on a not partitioned table is not possible
```

图 8-3 自动管理表分区时的错误信息截图

8.5 Zabbix 系统组件优化

前文我们所介绍的，针对操作系统和数据库的优化并不是 Zabbix 系统优化的最终目的和终点，我们优化的最终目的是为了 Zabbix 系统的各个组件高效和稳定地运行。因此，对 Zabbix 系统组件进行优化，也是我们对 Zabbix 系统优化的关键和最后的步骤。本节我们将介绍如何对 Zabbix 系统的各个组件进行优化。

我们知道，完整的 Zabbix 系统所包含的组件有 Zabbix 服务器、Zabbix 服务器代理、被监控设备代理、Web 前端组件和数据库系统等。在这些组件中，被监控设备代理是运行在被监控主机上的。一般来说，它的并发量和需要处理的数据量都不大，所以，它不是我们优化的重点目标。很显然，Zabbix 系统是属于组织内部的系统，所以，一般来说 Zabbix 系统 Web 前端的访问量和并发量也不会很大。因此，Web 前端组件的优化也不是我们优化的重点。最后，Zabbix 系统数据库的优化，我们在前面章节中已经做过详细介绍。这样，接下来需要重点优化的对象就只剩下 Zabbix 服务器和 Zabbix 服务器代理这两个组件了。而 Zabbix 服务器和 Zabbix 服务器代理这两个组件在配置上并没有太大的差异。因此，接下来，我们将以 Zabbix 服务器组件作为对象，详细介绍如何优化 Zabbix 服务器和 Zabbix 服务器代理组件。

8.5.1 Zabbix 服务器配置项说明

所谓对 Zabbix 服务器端组件进行优化，无非就是对 Zabbix 服务端组件的某些配置项的参数值进行调整，从而获得最大化的系统性能。既然要对 Zabbix 服务器组件的配置参数进行调整，那么首先就需要认识并弄懂这些配置项的作用和含义，从而使我们的调整和修改更具有针对性和目的性。然而，Zabbix 服务器端组件有多达一百多个配置项，因此，我们没有办法在这里对这一百多个配置项的含义和作用都作出介绍和说明。在这里，我们仅对可能影响系统性能的配置项的作用和含义作出简单的说明。如果读者需要完整地 Zabbix 服务器端组件的所有配置项的作用和含义，可阅读 Zabbix 官方所提供的用户手册。

下面这个配置项用于设置，用于存储被监控主机、监控项目和触发器等配置信息的共享内存的大小。因此，这个配置项应该设置的大小，与系统中所配置的监控项目数目、主机数和触发器数等配置信息有关。很显然，被监控的主机越多，则这个配置项的值就要相应地设置得大一些；反之，则可以设置小一些。然而，如果这个配置项的值被设置得过小，则会导致共享内存空间不够存放被监控元素的配置信息，从而使 Zabbix 服务器组件无法正常启动，并报如图 8-4 所示的错误信息。

```

24382:20140520:172550.455 _mmap_malloc: skipped 1 asked 27464 skip_min 2232 skip_max 2232
24382:20140520:172550.455 _mmap_malloc: skipped 2 asked 27464 skip_min 2232 skip_max 18296
24382:20140520:172550.455 [File:dbconfig.c:line:362] zbx_mmap_realloc(): out of memory (requested 27464 bytes)
24382:20140520:172550.455 [File:dbconfig.c:line:362] zbx_mmap_realloc(): please increase CacheSize configuration parameter
24382:20140520:172550.502 One child process died (PID:24382,exitcode/signal:255). Exiting ...

```

图 8-4 CacheSize 配置过小引起的错误信息截图

CacheSize

下面这个配置项用于指定，存储浮点型历史数据缓冲区的大小。默认值为 8MB，可设置的范围为 128K~2G，单位为字节。

HistoryCacheSize

下面这个配置项用于设置，存储文本型历史数据缓冲区的大小。默认值为 16MB，可设置的范围为 128K~2G，单位为字节。

HistoryTextCacheSize

下面这个配置项用于设置，存储趋势数据缓冲区的大小。默认值为 4MB，可设置的范围为 128K~2G，单位为字节。

TrendCacheSize

在前面章节中我们提到过，Zabbix 系统的 Web 前端组件和服务器端组件可以分开部署在不同的物理或逻辑服务器上。既然 Zabbix 系统的 Web 前端组件和服务器端组件可以分开部署，而且我们知道 Zabbix 系统 Web 前端组件主要是用 PHP 语言开发的，而服务器端组件主要是用 C/C++ 语言开发的，因此，就不难理解，当通过 Web 前端组件修改系统中的监控项目和被监控主机等配置信息时，Web 前端组件实际上是将修改后的配置信息存储到数据库中，然后，Zabbix 服务器端组件会定时读取数据库中的配置信息，并实施数据采集和监控。这样，就存在这么一个问题，那就是 Zabbix 服务器端组件多久去读取一次数据库中的配置信息呢？这个时长就由配置项 CacheUpdateFrequency 指定。该配置项的默认值是 60 秒，可以设置的范围是 1~3600 秒。也就是说，默认情况下，Zabbix 服务器端组件每隔 60 秒就会读取一次系统数据库中所配置的被监控主机和监控项目的配置信息，并将这些配置信息刷新到共享内存中。

应该不难理解，当 Zabbix 系统从数据库中读取配置信息并刷新到共享内存中时，或多或少是会占用一点数据库服务器和 Zabbix 服务器性能的。因此，如果所监控对象的配置不会被太频繁地更新，则尽可能地调大这个配置项的值，这样可以提高系统的性能。然而，如果将这个配置项的值设置得很大，则当通过 Web 前端页面修改了系统中被监控对象的配置时，将这些已更新的配置信息刷新到 Zabbix 服务器端就将会有相应的延迟。因此，这些已更新的配置信息就不会立即反映到系统的实际监控行为中。比如说将这个配置项的值设置为一个小时，也就是 3600 秒，那么在极端的情况下，当向 Zabbix 系统中添加一台新的被监控主机时，该被监控主机将在一个多小时后才会被采集监控数据并实施监控。

比较幸运的是，针对上述这种延迟刷新配置信息的情况，也可以通过手动执行下列命令来强制 Zabbix 服务器端进程刷新配置信息：

```

shell> /opt/zabbix/sbin/zabbix_server -R config_cache_reload -c /opt/zabbix
/etc/zabbix_server.conf

```

综上所述,从提升系统性能的角度来说,在监控配置更新不太频繁的 Zabbix 系统中,应尽可能地调大 CacheUpdateFrequency 配置项的值。

下面的这个配置项用于配置 MySQL 的 socket 文件的路径,默认值是/tmp/mysql.sock。该配置项只对后端数据库是 MySQL 的 Zabbix 系统有效。如果 Zabbix 服务器端组件和 MySQL 数据库处在同一台服务器上,则应将这个配置项的值设置成 MySQL 的 socket 文件实际路径,以便让 Zabbix 系统能够通过本地 socket 的方式连接数据库,从而提升系统的性能。

DBSocket

下面这个配置项用于设置输出日志信息的级别,默认值为 3,可设置值的范围为 0~4 的整数。数值越大,级别越低,也就意味着系统输出的日志信息就越多。毫无疑问,系统记录的日志信息越多对系统性能的影响就越大。因此,对于正式环境中的 Zabbix 系统,建议将该配置项的值设置为 0。

DebugLevel

下面这个配置项用于设置是否禁用“管家”进程。该配置项可以设置成 0 或 1,默认值是 0,即表示不禁用“管家”进程。而如果该配置项设置成 1,则表示禁用“管家”进程。前面介绍过,当启用自动表分区存储过程后,就可以禁用 Zabbix 系统中的“管家”进程,而通过自动表分区存储过程来管理 Zabbix 系统中的历史数据和趋势数据,这样做可以大大提高 Zabbix 系统的整体性能。

DisableHousekeeping

下面这个配置项用于设置“管家”进程的执行频率,单位是小时,可设置的范围是 1~24 小时,默认值为 1。如果 DisableHousekeeping 配置项被设置为 1,则下面这个配置项就会被忽略。与禁用“管家”进程类似,如果系统中“管家”进程需要开启,则调大这个配置项的值可以提升系统的性能。

HousekeepingFrequency

下面这个配置项用于设置系统记录数据库查询超过多少毫秒的查询语句。当系统执行的数据库查询语句其执行时间超过这个配置项所设置的时间时(即所谓的慢查询),系统就将相应的查询语句输出到日志文件中,并记录系统执行该查询语句所用的时间,单位是毫秒,可设置的范围为 0~3600000 毫秒,默认值为 0。当该配置项被设置成 0 时,则系统不记录慢查询的查询语句。这个配置项本身不会带来系统性能的提升,但是,通过开启该配置项,可以分析和发现系统中所执行的慢查询语句,并据此对数据库和 Zabbix 组件进行调优。

MaxHousekeeperDelete

下面这个配置项用于设置“管家”进程每次删除的最大记录数。可设置的范围是 0~1000000 条,默认值为 500。当该配置项的值设置为 0 时,则表示不限制每次删除历史数据的记录数。但是,很显然,这个配置项的值设置得越大,则可能导致“管家”进程在删除历史数据时所用的时间也就越长,同时也就可能导致对数据库锁表的时间越长。因此,该配置项的值不宜设置得过大。如果禁用“管家”进程,则该配置项会被忽略。

ProxyConfigFrequency

下面这个配置项用于设置, Zabbix 服务器端进程按照什么频率将监控对象配置信息的更新发送给 Zabbix 服务器代理端。该配置项只有在 Zabbix 服务器代理端工作在被动模式下时才有效,它的取值范围为 1~604800,默认值是 3600,单位是秒。与 CacheUpdateFrequency 配置项类似,如果系统中监控对象配置信息的更新不是非常频繁,则可以调大该配置项的值,从而提升系统的性能。然而,调高该配置项的值也就意味着,如果在系统中更新了被监控对象的配置信息,则配置信息将会以更慢的速度更新到 Zabbix 服务器代理端。

LogSlowQueries

下面这个配置项用于设置, Zabbix 服务器请求 Zabbix 服务器代理发送其所采集的监控数据的频率。可设置的范围为 1~3600, 默认值为 1, 单位是秒。很显然, 该配置项所设置的数值越小, 则表示 Zabbix 服务器代理发送其所采集的监控数据给 Zabbix 服务器的频率就越高, 从而对系统性能的影响也就越大。因此, 如果对监控数据的实时性要求不高, 则可以调高该配置项的值。该配置项只有在 Zabbix 服务器代理工作在被动模式下时才有效。

ProxyDataFrequency

当 Zabbix 系统给用户发送报警信息时, 如果没有发送成功, 则系统会重发。下面这个配置项用于设置, 系统发送未能发送成功的报警信息的频率。可取值范围为 5~3600, 默认值为 30, 单位是秒。

SenderFrequency

下面这个配置项用于设置, 系统中开启用于执行自动发现规则的进程数。可取值范围为 0~250, 默认值为 1。毫无疑问, 开启该类进程越多, 对系统资源的占用也就越大, 即使所开启的进程一直处于空闲状态也是要占用一定的系统资源的, 因此, 对于系统性能的影响也就越大。所以, 如果监控网络中设备变化不是很频繁, 则应调低这个配置项的值。而如果在系统中没有配置自动发现规则, 则应该将这个配置项的值设置为 0, 以免对系统造成不必要的资源占用。

StartDiscoverers

下面这个配置项用于设置, 系统中开启采集 Web 类监控项目监控数据的进程数。可取值范围为 0~1000, 默认值为 1。毫无疑问, 开启该类进程的数量越多, 对系统资源的占用也就越大, 即使所开启进程一直处于空闲状态, 也是要占用一定的系统资源的, 因此, 对系统性能的影响也就越大。所以, 如果系统中配置的 Web 类监控项目比较少, 则应尽可能地减小该配置项的值。上述这个规则同样也适用于接下来将要介绍的, 与系统启动进程有关的配置项。对于这些配置项, 如果系统中没有配置或者只是配置了少量相关类型的监控项目, 则应尽可能地调小相关配置项的值。

StartHTTPPollers

下面这个配置项用于设置, 系统中开启采集 IPMI 类监控项目监控数据的进程数。可取值范围为 0~1000, 默认值为 0。

StartIPMIPollers

下面这个配置项用于设置, 系统中开启采集 JMX 类监控项目监控数据的进程数。可取值范围为 0~1000, 默认值为 0。

StartJavaPollers

下面这个配置项用于设置, 系统用于调用 `fping` 命令工具的最大进程数。可取值范围为 0~1000, 默认值为 0。

StartPingers

下面这个配置项用于设置, 系统中开启用于采集不可到达主机数据的进程数。可取值范围为 0~1000, 默认值为 1。

StartPollersUnreachable

下面这个配置项用于设置, 系统中开启多少个进程, 用于请求被监控设备代理被动模式下所采集的监控数据, 可取值范围为 0~1000, 默认值为 1。

StartPollers

下面这个配置项用于设置, 系统中开启多少个进程, 用于 Zabbix 服务器与工作在被动模式下的 Zabbix 服务器代理进行通信。可取值范围为 0~250, 默认值为 0。

StartProxyPollers

下面这个配置项用于设置, 对于处于不可用状态的主机, 每隔多长时间检查一次它们的可

用状态。可取值范围为 1~3600，默认值为 60，单位是秒。很显然，这个配置项的值设置得越低，对系统性能的影响就越大。

UnavailableDelay

下面这个配置项用于设置，当主机的状态为不可到达时，每隔多长时间检查一次该主机新的状态。可取值范围为 1~3600，默认值为 15，单位是秒。

UnreachableDelay

下面这个配置项用于设置，当主机的状态处于不可到达状态多长时间时，系统就将其状态设置为不可用状态。可取值范围为 1~3600，默认值为 45，单位是秒。该配置项对于系统性能的影响不大，但是，它会影响主机在系统 Web 前端页面上状态的显示。

UnreachablePeriod

8.5.2 Zabbix 系统数据流分析

对于 Zabbix 系统来说，它所处理的主要对象其实就是各种监控数据。因此，当 Zabbix 系统遇到性能问题时，最直接的表现就是该采集或该处理的监控数据没有及时地采集到或者处理完成。然而，整个 Zabbix 系统由多个组件组成，而每个组件工作时又是由多种类型的进程来负责完成具体的任务的。因此，当 Zabbix 系统出现性能问题时，我们只有定位到是哪个组件，哪类进程出现了性能问题，我们才可以有针对性地进行优化。所以，分析和理解 Zabbix 系统中的数据流，对于我们分析和定位系统性能问题非常有帮助。图 8-5 所示即为 Zabbix 系统中数据流示意图。

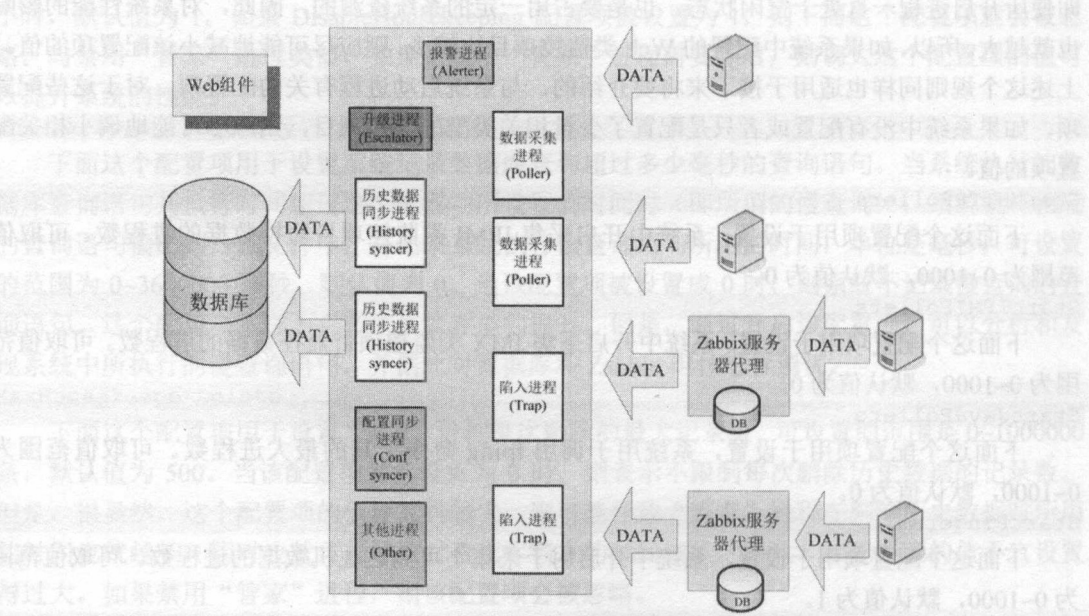


图 8-5 Zabbix 系统数据流图

由图 8-5 可以看出，当被监控项目不是由 Zabbix 服务器代理采集监控数据时，则 Zabbix 服务器的相关进程会根据监控项目的配置，定时采集相关的监控数据。但是，这些被监控数据采集进程所采集回来的监控数据，并不是由监控数据采集进程直接写入到数据库中，而是通过历史数据同步进程同步到数据库中。而如果被监控项目由 Zabbix 服务器代理采集监控数据，则这些被采集回来的数据会暂存在 Zabbix 服务器代理端的数据库中，然后 Zabbix 服务器代理端

相关进程会定期将这些数据,同步给服务器端相关进程,最后由服务器端的历史数据同步进程,将其写入到数据库中。需要注意的是,在 Zabbix 系统中,所有触发器表达式的计算,都是以系统数据库中所记录的数据为依据的。因此,如果在上述这些数据处理环节中某个环节出现了问题,比如数据处理不及时,则都可能会导致触发器被误触发,从而导致系统误发送报警信息。所以,当发现系统出现性能问题时,应该逐个环节排查。因为,只有逐个环节排查,才能更准确、更快速地定位出问题出在哪个环节。

8.5.3 Zabbix 系统性能问题表现

就像一个人得了感冒,往往会表现出咳嗽、流鼻涕等症状一样,当 Zabbix 系统出现性能问题时,往往也会有一些“症状”表现出来。我们只有熟悉和认识了这些“症状”,才能依据系统所表现出来的“症状”,判断系统是否出现了性能问题,进而有针对性地系统调优,或在必要的时候提升系统的硬件配置。

与其他应用系统一样,当 Zabbix 系统遇到性能瓶颈时,往往也会表现出系统 CPU 负载值长时间维持在较高水平上、系统中内存交换率很高、磁盘 I/O 速率很高等操作系统级的现象。对于这些操作系统级的现象,如何判断其是否是系统已经遇到性能瓶颈的现象,很多时候是需要靠长期积累的运维经验的,没有一套放之四海而皆准的理论公式可供我们参考。比如说磁盘 I/O 速率,因为磁盘的接口不同、服务器总线带宽千差万别以及磁盘本身性能的不同,我们很难说磁盘 I/O 速率达到多少值时,就预示着系统遇到了磁盘 I/O 性能问题。因此,对于 Zabbix 系统性能问题在操作系统级别上的表现现象,在这里不做深入的讨论,读者如果有需要,可以参考相关书籍。接下来,我们将着重讨论,当 Zabbix 系统遇到性能问题时,在 Zabbix 系统自身层面上的表现现象。

1. 系统每 ns 能处理多少条数据

通常,当评价一台 Web 服务器的性能时,往往通过其所能支持的最大并发量、吞吐率等指标来评价。然而,对于 Zabbix 服务器来说,如果再用并发量或吞吐率作为指标来评估它的性能就很不合适了。因为我们知道,Zabbix 系统作为组织内部的应用系统,其 Web 访问的并发量一般来说不会太高。而且,对于 Zabbix 系统来说,Web 组件服务并不是整个 Zabbix 系统中最占用系统资源的服务。

那么,对 Zabbix 服务器来说,该用什么指标来评价它的性能呢?对于 Zabbix 服务器,我们通常使用一个与并发量类似的指标——每秒处理新数据数(New Values Per Second, NVPS),来作为衡量 Zabbix 服务器性能的指标。

对于 Zabbix 系统中“每秒处理新数据数”这个指标,其实我们应该并不陌生。“Zabbix 状态”页面上“服务器性能(值/每秒)”项中所显示的内容就是该指标数据,如图 8-6 所示。

很显然,从主观意愿上来说,我们当然希望在相同的硬件配置情况下,我们的 Zabbix 系统每 ns 所能处理的数据越多越好。这样,就可以用尽可能低的硬配置,来监控尽可能多的主机和监控项目,从而尽可能地降低企业或单位的成本投入。然而,不管是什么样的应用系统,也不管我们怎么优化它,在同一硬件配置的情况下,其所能处理的最大数据量,或者说其所能达到的最高性能值总是有极限的。对于 Zabbix 系统来说也是一样的,对于某个固定的硬件配置,不管怎么优化,其所能监控的最多主机数总是有一个极限值的。

ZABBIX 状态		
参数	值	详细
Zabbix 服务器端运行中	No	10051
主机数量 (监控/未监控/模板)	145	116 / 0 / 29
项目数量 (监控中/已禁用/不支持)	5629	5507 / 42 / 80
触发器数量 (启用/关闭)[problem/unknown/ok]	1675	1604 / 71 [0 / 0 / 1604]
ZABBIX 系统用户数量	31	1
服务器性能 (值/每秒)	117.21	-
更新时间: 18:09:03		

图 8-6 Zabbix 状态信息

那么，这个极限值是多少呢？换句话说，当一台 Zabbix 服务器每 ns 所处理的新数据达到多少时，就认为这台 Zabbix 服务器已经基本上没有多少优化空间了，如果需要进一步提升系统性能，就需要升级硬件配置了呢？对于这个问题，没有一个非常严格的理论计算公式可供我们参考。但是，Zabbix ISA 公司的 CEO，也是 Zabbix 系统的开发者 Alexei • Vladishev，给我们提供了通用硬件配置环境下，Zabbix 系统最大可支持的每 ns 处理新数据数，可以作为我们的参考。

四核处理器（指物理内核，超线程出来的虚拟 CPU 内核不计），6G 内存，RAID10（带写入缓存的硬件 RAID）硬件配置的 Zabbix 系统，每分钟可以处理的新数据为大约 100 百万个，而每 ns 可处理的新数据大约是 15 000 个。当然，这个数值对于我们来说也只是一个参考，并不是严格意义上的“标准值”。这是因为，一方面，我们的 Zabbix 服务器的硬件配置情况不总是与上面案例中服务器的硬件配置情况完全一致；另一方面，操作系统的版本、数据库的版本不同，对整个 Zabbix 系统的性能也是有比较大的影响的；再者就是监控项目的数据类型，其对于 Zabbix 系统的性能也是有比较大的影响的。

你或许还记得，我们在前面章节中，介绍 Zabbix 状态的“服务器性能（值/每秒）”这个项目时提到过，系统中这个项目所显示的数值，并不是系统处理数据的实际值，而是 Zabbix 系统根据我们所配置的监控项目和监控项目所配置的监控数据采集间隔估算出来的。更重要的是，Zabbix 系统计算这个项目数据所依据的监控项目，并不是系统中所配置的全部监控项目，而只是被动类型的监控项目。所以，用服务器每 ns 能处理多少个新数据这一指标，来衡量 Zabbix 服务器的性能，只能说在一定的范围内它具有参考意义，它并不能成为我们衡量 Zabbix 服务器性能的唯一指标。

2. Zabbix 系统队列可以反映系统性能问题

当依次选择菜单项“高级配置”→“队列”时，系统将显示如图 8-7 所示的系统队列列表信息。

理想情况下，当打开图 8-7 所示的这个页面时，如果页面上所有数据行都是绿色的，则意味着队列中没有被监控项目。在这种情况下，同样也意味着，所有的监控项目的监控数据的采集都没有被延迟，即所有的监控项目都按照配置计划采集监控数据，并被更新到系统数据库中。然而，当服务器遇到某种性能瓶颈问题时，就会有一些监控项目的监控数据被延时采集，或被延时更新到数据库中。因此，系统队列中被延时的监控项目数越多，被延时的时间越长，通常也就意味着系统遇到的性能问题越严重。当然，也有一些其他原因，比如，Zabbix 服务器和被监控主机之间的网络通信问题，被监控设备自己遇到了性能问题等，也会导致队列中延迟采集监控数据的监控项目数增加，且延迟时间变长。因此，在通过 Zabbix 系统的队列信息判断 Zabbix

服务器是否遇到性能瓶颈时，应尽可能地排除这些因素的影响。

已更新的项目队列						
项目	5秒	10秒	30秒	1分	5分	10分钟以上
Zabbix agent	0	0	0	0	0	0
Zabbix agent(主动方式)	0	0	0	0	0	0
Simple check	0	0	0	0	0	0
SNMPv1 agent	0	0	0	0	0	0
SNMPv2 agent	0	0	0	0	4	4
SNMPv3 agent	0	0	0	0	0	0
Zabbix internal	0	0	0	0	5	0

图 8-7 Zabbix 系统队列信息列表

通过“队列”页面，我们不但可以判断 Zabbix 服务器是否遇到性能瓶颈问题，而且因为“队列”页面上所显示的延迟采集监控数据的监控项目数，是按监控数据采集的不同方法分别进行分类统计与显示的。所以，通过这个页面，可以初步判断是哪类 Zabbix 进程可能遇到了性能瓶颈问题，从而为系统进一步调优提供依据和参考。

3. Zabbix 系统性能问题的其他表现

除了上面所述的，通过 Zabbix 服务器的性能值和队列信息，可以判断 Zabbix 服务器是否出现了性能问题外，当 Zabbix 系统出现如下这些现象时，也往往预示着 Zabbix 服务器可能遇到了性能瓶颈。

❑ **数据图有断图。**当通过 Web 前端组件查看监控项目的数据图时发现大量的断图，特别是不同主机上的监控项目几乎在同一个时间段都出现了断图现象时，则预示着 Zabbix 服务器很可能是遇到了性能瓶颈。这是因为，当 Zabbix 服务器遇到性能瓶颈时，监控数据可能就无法及时地被采集到或者无法及时地被写入到数据库中，因而通过 Web 前端组件查看数据图时就会有大量的断图现象。

当然，如果网络的稳定性出现了问题，也会导致数据图产生断图现象；同时，如果被监控主机自身出现性能问题，那么它就可能无法及时响应 Zabbix 服务器采集监控数据的请求，同样也会导致数据图有中断的现象。

❑ **触发器误触发现象明显增加。**当 Zabbix 服务器遇到性能瓶颈时，可能会带来某些触发器被误触发，特别是触发器计算表达式中使用了 `nodata()` 函数的触发器。这其实跟数据图中有断图是一个道理，当监控数据没有被及时采集到，或者未及时写入到数据库中时，则可能导致 `nodata()` 函数的返回值为“真”，从而导致计算表达式中调用了该函数的触发器被触发。

❑ **Web 前端页面打开缓慢。**当打开 Web 前端页面非常缓慢时，则预示着 Zabbix 服务器的某些资源很可能接近被耗尽或者数据库系统遇到了较严重的性能瓶颈。而如果将 Web 前端组件单独部署在独立的物理服务器上，且以前打开 Web 前端页面正常，突然某一天打开它非常缓慢，则 Zabbix 系统的数据库遇到了性能瓶颈的可能性非常大。

8.5.4 Zabbix 系统内部状态监控

虽然上述的“队列”页面和“服务器性能”值可以在某种程度上反映出 Zabbix 系统的当前工作状态。但是，仅依靠这些数据，在很多时候如果 Zabbix 服务器出现性能问题，仍然无法准确地定位出现性能问题的具体地方。因此，要想在 Zabbix 服务器出现性能问题时，能够快速而准确地定位出 Zabbix 系统出现性能瓶颈的地方，就需要对 Zabbix 系统中各种进程和组件的工作状态进行实时的监控。

我们知道，Zabbix 系统内部数据采集方法就是用于采集 Zabbix 系统内部数据的。所以，通过该方法就可以实现实时采集和监控 Zabbix 系统内部状态数据。

在众多可采集和监控的 Zabbix 系统内部状态数据中，对于我们系统调优有帮助的数据主要有三类：一类是反映系统中各进程工作忙闲状态的状态数据，它的关键字形式是：`zabbix[process,<type>,<mode>,<state>]`（关于这个关键字各参数的含义及使用它的方法，请参阅本书前面相关章节中的介绍）。通过它，可以实时采集和监控 Zabbix 系统中各个进程的忙闲状态，图 8-8 所示即为所对应的数据图。

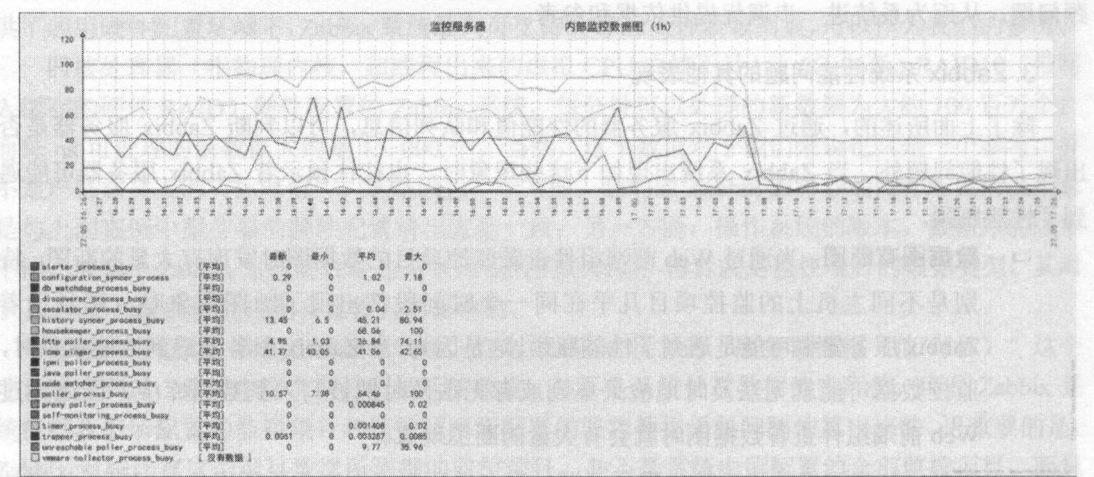


图 8-8 Zabbix 系统进程状态数据图

通过图 8-8 这种反映 Zabbix 服务器各进程状态的数据图，就可以很容易地发现哪些进程经常性处于空闲状态，而又有哪些进程经常性处于繁忙状态。对于那些经常性或者长期处于空闲状态的进程，可以通过调整配置文件中相关配置项的值，来减少服务器开启该类进程的数量，从而减少因为开启了过多该类进程而造成的对服务器资源的不必要占用。而对于经常性地处于繁忙状态的进程，同样也可以通过调整配置文件中相关配置项的值，来增加服务器开启该类进程的数量，从而提高系统处理相关数据的速度，进而提升系统整体的性能。

需要监控的第二类 Zabbix 系统内部数据是，各类缓冲区的使用情况。这类监控项目的关键字是 `zabbix[wcache,<cache>,<mode>]`（关于这个关键字各参数的含义及使用它的方法，请参阅本书前面相关章节中的介绍）。通过它，可以实时采集和监控 Zabbix 系统中各类缓冲区的使用情况，如图 8-9 所示。

通过图 8-9 所示的数据图，可以实时监控系统中各类缓冲区的使用情况。当发现某类缓冲区设置过小时，我们应及时调整相关配置项的值，以免因为缓冲区设置过小而影响整个系统的性能。我们建议，一般如果缓冲区的空闲空间长期维持在小于 20%的水平，则应该考虑调大对

应缓冲区的大小。类似的，如果某类缓冲区长期保持在 80%以上的空闲空间，则就应该考虑减小该缓冲区的大小。当然，上述的 20%和 80%值为经验总结，你也可以根据你自己的经验，来判断何时应该调大或调小缓冲区的大小。

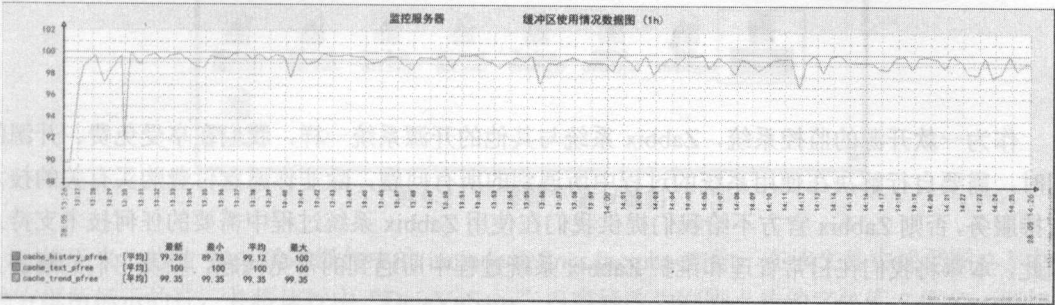


图 8-9 Zabbix 系统缓冲区空闲空间数据图

第三类需要监控的内部数据是队列长度、服务器性能数据、系统不支持的监控项目数等。通过对这些数据的监控，可以及时发现系统中可能存在的性能问题。例如我们知道，当系统中存在大量不被支持的监控项目时，无疑会对系统的性能产生很大的影响。所以，也应该对这类内部数据进行监控，并设置相应的触发器。

8.6 本章小结

本章首先对 Zabbix 系统的特点作了一些分析，从而我们知道 Zabbix 系统是属于一种偏写入型的重数据库的应用系统。在此基础上，讨论了在对 Zabbix 系统进行优化时应遵循的一些原则，并从操作系统、数据库和 Zabbix 系统组件等三个层面讨论了如何针对 Zabbix 系统进行调优。

笔者认为，针对任何一种系统进行调优都是一个系统化的、持续化的过程，很多调优过程中遇到的问题，都是需要具体问题具体分析的。因此，在这里，笔者并没有像其他许多书籍那样提供一套“标准”的配置文件给你，而是讨论了应该从哪些方面对 Zabbix 系统进行优化，具体某个配置项应该设置成什么数值，那应该根据系统的硬件配置、系统中被监控主机和监控项目的配置进行设置，并不断调整和优化。

第 9 章 常见问题及使用技巧

作为一款开源的监控系统，Zabbix 系统与其他的开源系统一样，我们在享受免费、开源的同时，需要自行解决在使用系统的过程中所遇到的所有问题，除非你愿意付费购买有关的技术支持服务，否则 Zabbix 官方不给我们提供我们在 Zabbix 系统过程中需要的任何技术支持。因此，本章将我们在日常管理和维护 Zabbix 系统过程中所遇到的常见问题，以及它们的解决方法进行了收集和整理，供大家参考。

同时，本章也将我们在日常管理和维护 Zabbix 系统过程中可能使用到的小技巧和小经验进行了收集和整理，供大家参考。

9.1 为什么数据图中的中文显示为乱码

当你按照本书第 1 章所介绍的方法安装部署完 Zabbix 系统后，你可能会发现当数据图或拓扑图中包含中文信息时，所有的中文字符都显示成了乱码，如图 9-1 所示。

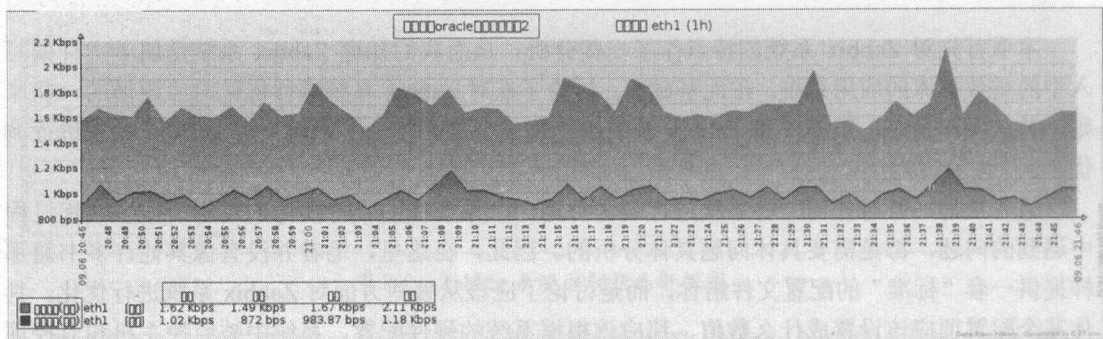


图 9-1 数据图中的中文显示为乱码

从图 9-1 可以看出，红色方框标示的地方本来显示的应该是中文字符，可是在图中都显示成乱码了，为什么会出现这种情况呢？出现这个问题的主要原因是因为，在 Zabbix 系统的 Web 前端组件中没有包含可用的中文字体库文件。所以，要解决这个问题，只需在 Web 前端组件中添加上某个中文字体库文件即可，具体的操作步骤如下：

在任何一台 Windows 系统的主机上依次选择菜单项“开始”→“设置”→“控制面板”，在“控制面板”窗口中双击“字体”图标，系统将打开“字体”窗口，如图 9-2 所示。

从图 9-2 所示的“字体”窗口中选中需要的中文字体，例如“楷体”，将该字体文件（“楷体”字体文件名为 simkai.ttf）复制并上传到 Zabbix 系统 Web 前端组件部署路径的 fonts/目录下（如果是按照本书第 1 章介绍的方法安装部署 Zabbix 系统 Web 前端组件的，则其部署路径为 /data/website/zabbix）。

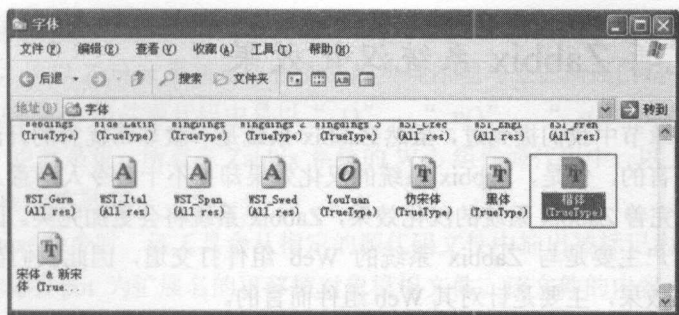


图 9-2 “字体”窗口截图

接下来，修改 Web 前端组件中的 `include/defines.inc.php` 文件，在该文件中找到图 9-3 中红色方框所标示的行，并将该行中“DejaVuSans”内容修改为刚刚上传的字体库文件名。例如，上传的字体库文件是楷体字体库文件，则该处的内容应修改为“simkai”。注意，该处的内容应不包括扩展名的字体库文件名，且如果所上传的字体库文件的扩展名是大写的 TTF，则需要将其扩展名修改为小写的 ttf。

```
define('ZEM_MIN_PERIOD', 3600); // 1 hour
define('ZEM_MAX_PERIOD', 63072000); // the maximum period for the time bar control. 72 years (2 * 365 * 86400)
define('ZEM_PERIOD_DEFAULT', 3600); // 1 hour
define('ZEM_WIDGET_ROWS', 20);
define('ZEM_FONTPATH', realpath('font')); // where to search for font (GD > 2.0.18)
define('ZEM_GRAPH_FONT_NAME', 'DejaVuSans'); // font file name
define('ZEM_GRAPH_LEGEND_HEIGHT', 120); // when graph height is less than this value, some legend will not show up
define('ZEM_SCRIPT_TIMEOUT', 60); // in seconds
```

图 9-3 `include/defines.inc.php` 文件内容截图

经过上述修改，我们再来看数据图，以验证所作的修改是否正确。如果上述各步骤的操作均正确，则数据图和拓扑图上的汉字将不会再显示成乱码，如图 9-4 所示。

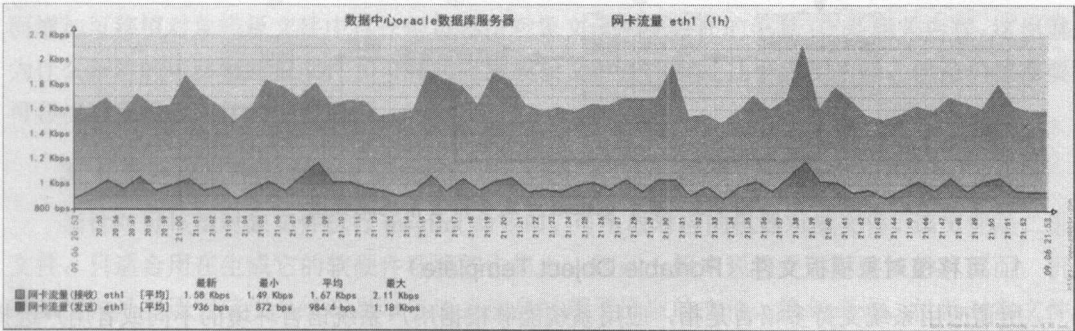


图 9-4 网卡流量数据图

- 提示：1. 从 Windows 的“字体”窗口复制字体库文件时，你可能会发现通过菜单无法复制和粘贴，需要通过 Ctrl+C 和 Ctrl+V 组合键来复制字体库文件。
2. 你也可以直接用从 Windows 系统中复制过来的字体库文件替换掉 Web 前端组件部署路径 `fonts/` 目录下的 `DejaVuSans.ttf` 文件，这样就可以不修改 `include/defines.inc.php` 文件内容了。

9.2 如何完善 Zabbix 系统汉化效果

在本书的前面章节中我们提到过，虽然 Zabbix 系统是一款非常优秀的开源监控系统，且其本身也是支持多语言的，但是，Zabbix 系统的汉化效果却并不十分令人满意。因此，在有能力的情况下，如果能完善 Zabbix 系统的汉化效果，Zabbix 系统将会更加完美。因为在日常的使用和维护过程中，用户主要是与 Zabbix 系统的 Web 组件打交道，因此我们在这里所说的完善 Zabbix 系统的汉化效果，主要是针对其 Web 组件而言的。

Zabbix 系统多语言支持是通过 gettext 软件包来实现的，这个软件包中包含多个工具。当通过 gettext 软件包实现应用系统的多语言支持时，系统需要遵循一定的规范。首先我们来看看 Zabbix 系统是怎样通过 gettext 软件包来实现多语言支持的。

9.2.1 基于 gettext 多语言支持系统的开发流程

详细全面地介绍开发支持多语言应用系统所需的相关知识，显然已经超出了本书的范围，但是对基于 gettext 多语言支持的应用系统开发流程作一个简单的了解，对更好地理解接下来将要介绍的汉化 Zabbix 系统的操作步骤，以及为何要如此操作有很大的帮助。因此，接下来我们简单地介绍一下基于 gettext 开发多语言支持应用系统的流程，其流程如图 9-5 所示。

从图 9-5 所示的过程可以看出，在整个开发过程中，分别在第③、第④和第⑥步有新的不同类型的文件生成，这些文件的作用和含义如下所述。

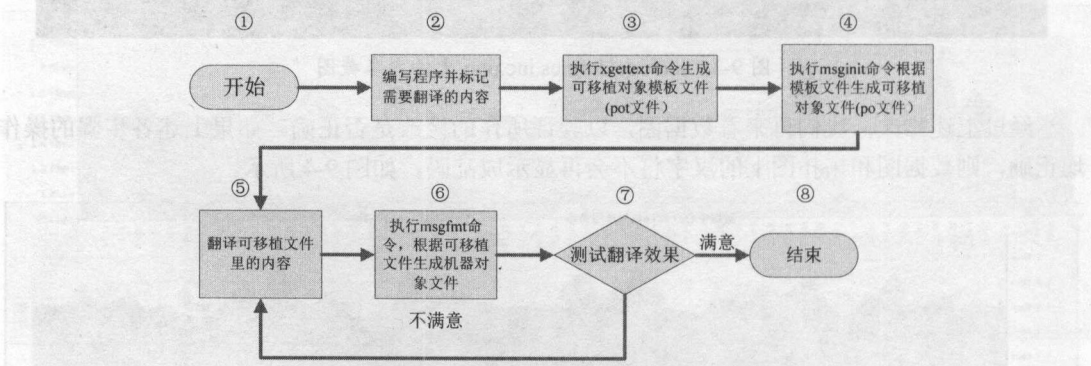


图 9-5 基于 gettext 开发多语言系统流程图

1. 可移植对象模板文件（Portable Object Template）

所谓应用系统支持多语言是指，应用系统能够根据用户系统语言环境的不同或者用户选择的不同，使用不同的语种向用户显示系统信息的功能。因此，我们不难理解，在支持多语言的应用系统中，需要在源代码里将那些可能需要使用不同种语言显示的内容使用特定格式的标识符标识出来，以便在用户选择不同的语种时，系统能够使用指定的语言显示相应的信息。否则，应用系统将无法识别和判断哪些信息内容是需要依据用户所选择语种的不同进行处理的，而哪些信息内容又是不需要处理的。那些在应用系统源代码中被特定格式标识符标识出来的内容就是信息 ID，在应用系统运行时，其会以信息 ID 为参数调用 gettext 函数，从而获取用户所指定语种的信息内容。或许你会说，不是所有支持多语言的应用系统都是通过上述方式来实现的。是的，应用系统实现多语言支持的方法有多种，但是所有通过 gettext 软件包实现多语言支持的

应用系统，都是通过上述方式实现的。而前面介绍过，Zabbix 系统的多语言支持就是通过 gettext 软件包来实现的，所以 Zabbix 系统多语言支持的实现原理与上面所述的是一致的。

Zabbix 系统的 Web 组件源代码中是以 “_()”、“_n()”、“_s()”和 “_xn()”等标识符来标识信息 ID 的，也就是说，所有在 Zabbix 系统的 Web 组件源代码中，以上述标识符所标识起来的内容，都被视为信息 ID。

当执行 xgettext 命令时，该工具会从指定的源代码文件中抽出被标记为信息 ID 的内容，并按一定的格式生成以 pot 为扩展名的可移植对象模板文件。该文件的内容格式与接下来将要介绍的可移植对象文件的内容格式是一样的，只是它是系统中所有被支持语言所对应的可移植对象文件的通用模板，且其内容中需要填写翻译内容的地方均是留空的。

2. 可移植对象文件(Portable Object)

在系统中生成可移植对象模板文件后执行 msginit 命令，该工具会依据可移植对象模板文件的内容，针对系统所支持的每一个语种生成相对应的以 po 为扩展名的可移植对象文件。从这类文件的类型名中可以看出，该类文件是可以在不同平台和不同系统之间进行移植的，在汉化过程中我们所需要修改的文件也就是这种类型的文件。当使用 msginit 命令新生成可移植对象文件时，其内容与可移植对象模板文件的内容几乎完全相同。这样就带来了一个问题，那就是如果因为应用系统的源代码做了修改，比如添加了新的信息 ID 或删除了某些信息 ID，而我们仍然使用 msginit 命令来生成或更新可移植对象文件，那么新生成的可移植对象文件中需要填写翻译内容的地方都将会变成空字符串。换句话说，此时我们又需要从头翻译所有内容，显然这是非常让人难以接受的事情。因此，当因为应用系统的源代码做了修改，而需要更新可移植文件时，我们不再使用 msginit 命令工具，而是使用 msgmerge 命令来更新它们。msgmerge 工具会对比可移植对象模板文件和可移植对象文件中的内容，并依据它们之间的差异来修改和更新可移植对象文件中的内容，即在可移植文件中删除可移植对象模板文件中不存在的信息 ID 及相关内容，而增加可移植对象模板文件中存在但可移植对象文件中不存在的信息 ID 及相关内容。这也就是为什么新建的可移植对象文件和可移植对象模板文件在内容上几乎完全一样，但是仍然需要有可移植对象模板文件的主要原因。

3. 机器对象文件(Machine Object)

当针对可移植对象文件执行 msgfmt 命令时，可生成对应的机器对象文件。该文件是二进制文件，只适合用在生成它的软硬件环境的主机上，因此一般来说它是不可移植的。然而，机器对象文件是多语言应用系统在显示多语言信息时需要读取的文件，换句话说，机器对象文件才是多语言应用系统需要的最终文件，其他的如可移植对象文件和可移植对象模板文件都为最终生成机器对象文件而产生，它们是中间文件。

9.2.2 可移植对象文件格式说明

前面提到过，当要完善 Zabbix 系统的汉化效果时，需要修改的文件就是可移植对象文件。因此，需要首先对该类型文件内容的格式作一些了解，否则将无法对它们进行编辑和修改，也就更谈不上完善 Zabbix 系统的汉化效果了。

首先来看看如图 9-6 所示的这段从 Zabbix 系统的可移植对象文件中截取出来的信息。

```

"Project-Id-Version: Zabbix 2.1\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2014-03-31 13:00+0300\n"
"PO-Revision-Date: 2013-08-25 15:11+0200\n"
"Last-Translator: solutionware <shiys@solutionware.com.cn>\n"
"Language-Team: Zabbix <info@zabbix.com>\n"
"Language: zh_CN\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=1; plural=0;\n"
"X-Generator: Poedit 2.1.6\n"
"X-Poedit-Basepath: ../../\n"

#: api/classes/CMaintenance.php:387 api/classes/CMaintenance.php:392
#: api/classes/CMaintenance.php:500 api/classes/CMaintenance.php:505
#: maintenance.php:141 maintenance.php:153 services.php:184 services.php:198
#, c-format
msgid "\"%s\" must be between 1970.01.01 and 2038.01.18."
msgstr "\"%s\" 必须位于 1970.01.01 和 2038.01.18."

#: include/profiles.inc.php:320
msgid "\"Display OK triggers\" needs to be \"0\" or a positive integer."
msgstr "\"显示 正常的 触发器\" 必须为 \"0\" 或一个正整数."

#: include/profiles.inc.php:325
msgid "\"Triggers blink on status change\" needs to be \"0\" or a positive integer."
msgstr "\"触发器闪烁状态改变时\" 必须为 \"0\" 或一个正整数."

#: include/func.inc.php:465
#, c-format
msgid "%1$d day"
msgid_plural "%1$d days"
msgstr[0] ""

```

图 9-6 可移植对象文件内容截图

由图 9-6 可以看出，整个可移植对象文件内容是由一行行空白行分隔出来的实体组成的。其中，第一行空白行以上的内容定义了可移植对象文件的版本、创建时间、最后修改时间、翻译者以及文件的编码等内容，一般来说这部分内容我们无须修改。接下来的每个被单行空白行分隔出来的实体内容都是类似的，其中：

以#开头的行是注释行，用于说明其下行需要翻译的内容出自哪个（些）源代码文件，以及在对文件中的位置。这些注释行如果被删除掉，则不会影响可移植对象文件生成机器对象文件，但是有了这些注释行有助于我们理解需要翻译的信息内容。

类似的，以#开头的行也是注释行，用以说明在接下来需要翻译的信息行中包含哪种格式的占位符，例如 C 语言格式的占位符，即形如%s、%d 等形式的占位符。虽然以#开头的行同样也为注释行，但是这类注释行有一些特殊性，因为在执行 msgfmt 命令将可移植对象文件生成机器对象文件时，它们不会被当作普通的注释行而被完全忽略掉。当 msgfmt 命令将可移植对象文件生成机器对象文件时，它会读取以#开头的注释行内容，并使用它们为用户生成更良好的诊断信息。

以关键字 msgid 开头的行所指示的内容即为信息的 ID，也就是我们在前文所述的，在 Web 组件的源代码中以“_0”、“_n0”、“_s0”、“_xn0”等标识符所标识的内容。或许说它们是信息 ID 也不十分贴切，因为它们其实就是英语版的完整信息内容。换句话说，当将 Zabbix 系统的 Web 前端组件语言设置成英语，或者虽然我们将其设置成中文，但是某些信息尚未被翻译成中文，则系统在显示时将以关键字 msgid 所指定的内容显示在 Web 前端页面上。

如前文所述，信息 ID 是由 msginit 或 msgmerge 命令直接从程序源代码文件中提取的，所以一般来说我们在对 Zabbix 系统进行汉化效果完善时，不需要修改信息 ID 部分的内容。

以关键字 msgstr 开头的行所指定的内容，即为已翻译的信息内容，它也就是我们在完善 Zabbix 系统汉化效果时需要补充和完善的主要内容。

以关键字 msgid_plural 开头的行所指定的内容，为信息 ID 的复数形式。我们知道在很多语

言中，比如英语，同一句话的单复数表达形式不完全一样。这样，某些信息在显示时就需要区分单复数形式，因此在可移植对象文件中也就有了 `msgid_plural` 关键字所指定的内容。至于信息的单复数形式，在源代码层面是如何区分和分别提取的（其实在源代码层是通过不同的函数来提取的），这并不是我们关注的重点。对于完善 Zabbix 系统汉化效果而言，只需知道关键字 `msgid_plural` 所指定的内容是信息的复数形式，并做相应的汉化即可。

类似的，以关键字 `msgstr[0]` 开头的行，为已翻译信息内容的复数形式。在某些语言中，同样一句话，所表示的数字不同，其复数有多种形式时，可以用 `msgstr[1]`、`msgstr[2]`、`msgstr[3]` 等数组形式的关键字来表示。

9.2.3 Zabbix 系统汉化效果完善

前面介绍了可移植对象文件的格式，接下来介绍如何具体地汉化 Zabbix 系统。当使用 `gettext` 软件包来开发多语言支持的应用系统时，其可移植对象文件和机器对象文件的存放路径需要遵循一定的规范。对于 Zabbix 系统的 Web 组件来说，这些文件都存放在源码包的 `<Web_root>/local/<语种名称简写>/LC_MESSAGES/` 目录下。其中“语种名称简写”需要遵循“ISO 639”规范。如果某种大的语种有多个不同国家或地区使用习惯，则“语种名称简写”就需要以“语种名称简写_国家或地区名称简写”的形式书写。例如同样是中文，简体中文的语种名称就简写成“zh_CN”，而中国台湾地区使用的繁体中文名称则简写成“zh_TW”。而在 Zabbix 系统中，可移植对象文件名和机器对象文件名分别为“frontend.po”和“frontend.mo”。因此，在完善 Zabbix 系统汉化效果时，只需翻译 `<Web_root>/local/<语种名称简写>/LC_MESSAGES/frontend.po` 文件里的内容，并依照翻译后的可移植对象文件（frontend.po）重新生成机器对象文件（frontend.mo）即可，图 9-7 所示即为 Zabbix 系统 Web 组件源码包中 local 目录下的目录和文件列表。

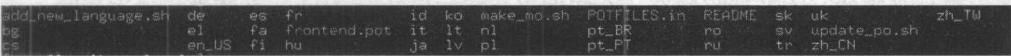


图 9-7 local 目录下目录和文件列表

由图 9-7 可以看出，在 local 目录下除了有上文所述的“语种名称简写”目录外，还有三个 Shell 脚本程序文件，这三个脚本程序的作用如下：

- ❑ `add_new_language.sh`。如果需要让 Zabbix 系统支持一种全新的语言，则需要执行 `add_new_language.sh` 脚本，它会依据可移植对象模板文件（frontend.pot 文件）中的内容，生成指定语种的可移植对象文件。需要注意的是，在全新下载的 Zabbix 系统源码包中，可能不包含可移植对象模板文件，此时如果尝试执行 `add_new_language.sh` 脚本程序，则脚本程序会报找不到可移植对象模板文件错误。此时，需要首先执行同一目录下的 `update_po.sh` 脚本程序。该脚本程序会扫描所有 Zabbix 系统 Web 组件源程序文件，并提取其中被标记为信息 ID 的内容，生成可移植对象模板文件。
- ❑ `make_mo.sh`。`make_mo.sh` 脚本程序会自动查找 local 目录下，所有子目录下的可移植对象文件，并依据它们的内容生成各自对应的机器对象文件。因此，如果修改了可移植对象文件中的内容，则需要执行该脚本程序以更新机器对象文件。否则，所作的任何修改都不会起作用。
- ❑ `update_po.sh`。由这个脚本程序的文件名也不难猜出，执行该脚本程序将会更新可移植对象文件内容。当修改了 Web 组件的源程序，比如向其中某个源程序文件中添加了新

的信息 ID，或者删除了某个源程序文件中某些信息 ID，都需要执行这个脚本程序，以便根据系统的源程序文件内容更新可移植对象文件中的内容。另外，执行 `update_po.sh` 脚本程序除了能更新所有语种对应的可移植对象文件内容外，还会更新或创建可移植对象模板文件。

综上所述，如果需要完善 Zabbix 系统的汉化效果，其操作过程为：首先执行 `update_po.sh` 脚本程序，以更新系统中所有可移植对象文件内容。接下来，完善 `local/zh_CN/LC_MESSAGES/frontend.po` 文件中的翻译内容。最后，执行 `make_mo.sh` 脚本程序，以更新机器对象文件。

由于篇幅的限制，这里没有办法提供完整的已经翻译好的可移植对象文件内容。况且，Zabbix 系统的版本不同其 Web 组件的源程序也不完全一样，所以，即使我们能够提供一份完整的已经翻译好的可移植对象文件内容，也可能无法适用于你的系统。因此，这里我们仅能介绍完善 Zabbix 系统汉化效果的操作过程和方法，具体对可移植对象文件内容的翻译还需要读者自行完成。

9.3 如何批量添加图表

对于 Zabbix 系统，相信绝大多数使用过它的用户都认可其功能强大、绘图能力强且美观、易用性好等特点。但是，笔者曾在某技术论坛上看到，有用户抱怨 Zabbix 系统使用起来烦琐，缺乏灵活性等。笔者认为，这些抱怨或许有一定的道理，但是又不完全正确。我们且不说应用系统的易用性和灵活性在很多时候就像鱼和熊掌不可兼得一样，是一个矛盾的问题，单从用户对应用系统的功能需求方面来说，我们认为几乎没有任何一款开源的应用系统能够完全满足其所有用户的所有需求。因此，在很多时候针对用户的特殊需求，对 Zabbix 系统进行二次开发和功能完善是必需的。本节就将结合一个实例来介绍如何针对 Zabbix 系统进行二次开发。

9.3.1 基本功能说明

我们知道，在 Zabbix 系统中有一种功能叫做图表（Screen）。通过图表，可以将系统中诸如简单数据图、数据图、拓扑图等众多的信息放到一张表格中，以供大家在必要的时候查阅。这个功能在某些时候是非常有用的，比如当对某个系统进行调优时，我们可能需要频繁地查看诸如 CPU 负载情况数据图、网卡流量数据图、内存使用率数据图等，并对它们在同一时刻的数据和趋势情况进行比较。虽然这些数据图在系统中可能都是存在的，而且也可以非常方便地对它们进行查看，但是通常需要通过不同的页面来查看，无法将它们同时显示在同一个页面上，以方便进行对比查看。

然而我们知道，通过 Web 组件的自身功能我们每次只能创建一张图表。这样，如果希望一次针对大量主机上数据图或简单数据图分别创建图表那将是一份苦差事。例如，假如我们有几十台或者更多的主机，它们各自都有 CPU 负载情况数据图、网卡流量数据图、内存使用率数据图和磁盘 IO 速率数据图等，现在希望针对每台主机，以上述这些种类的数据图作为资源创建以各主机的主机名命名的图表，以便能够在同一个页面上查看每台主机上的所有数据图，从而进行数据对比和分析。针对上述这个需求，如果使用 Zabbix 系统 Web 组件自身的功能，通过手工逐个创建相应的图表，不但费时费力，创建起来非常枯燥乏味，而且还非常容易出错。下面将要介绍的针对 Zabbix 系统的二次开发实例——批量添加图表功能就是为了解决这个问题的，

其基本功能为：

- 根据用户输入的主机名列表，在系统中创建以列表中每台主机名命名的图表。
- 根据用户输入的资源关键字列表，在系统中针对主机名列表中所指定的主机，搜索数据图或简单数据图名称（对于简单数据图而言，它的名称就是对应监控项目的名称）中包含关键字的数据图或简单数据图。
- 将搜索到的数据图或简单数据图作为资源添加到相应图表中。

很显然，要完成上述功能，需要开发相应的表单页面，该功能的表单页面效果图如图 9-8 所示。

批量添加图表

图表列数：

图表资源类型：

主机IP地址列表：

资源名称列表：

宽：

高：

横向对齐：☐ 左侧 ☒ 居中 ☐ 右侧

纵向对齐：☐ 顶部 ☒ 居中 ☐ 底部

动态项目：☐

提交查询内容

图 9-8 批量添加图表表单页面截图

9.3.2 数据表关系分析

不难理解，要实现上述的批量添加图表功能，我们的程序要做的工作就是在 Zabbix 系统数据库中查找、更新和插入相关数据记录。因此，在开始编写程序前，需要对程序要操作的数据表以及它们之间的关系进行必要的分析和研究，以确定在程序中要对哪些数据表进行操作以及怎样操作。通过分析，我们在程序中需要操作的 Zabbix 系统数据表以及它们之间的关系如图 9-10 所示。

由图 9-9 不难看出，批量添加图表程序需要操作的数据表有：

1. 图表表(screens)
该数据表中存储的是系统中的所有图表信息，包括图表 ID (screenid)、名称 (name)、列数 (hsize) 和行数 (vsize) 等字段。其中，图表 ID 字段是图表表的主键。
2. 图表项目关系表 (screens_items)
图表项目关系表的字段比较多，限于篇幅的原因，在图 9-9 中只列出可能需要使用到的字段，其他字段均略去了。类似的，对于主机表 (hosts)、数据图表 (graphs) 和监控项目表 (items) 等，我们在图 9-9 中也仅列出了可能需要使用到的对应表的字段，其他字段均略去。对此，在后续的叙述中不再一一说明。

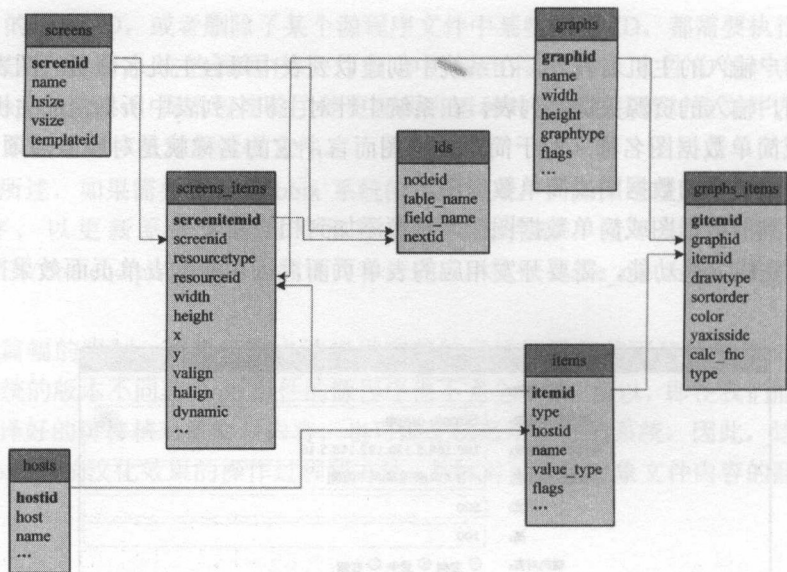


图 9-9 数据表关系图

我们知道，一张图表中可以放置多个资源项目，同样，任意一个资源项目也可以被放置在不同的图表中。因此，图表和资源项目之间是多对多的关系，这样就需要有一张数据库表来记录图表和资源项目之间的关系。在 Zabbix 系统中，图表项目关系表（`screens_items`）即是用于记录这种关系的数据库表。其中，该表中的图表项目关系 ID 字段（`screenitemid`）是该表的主键，而图表 ID 字段（`screenid`）是图表表（`screens`）的外键。我们还会使用到该表中的资源类型字段（`resourcetype`），该字段用于记录图表中某个单元格所放置资源的类型，它的取值范围为 0~16 之间的整数，对应于图表中可以放置的 16 种类型的资源，它们之间的对应关系是在 Web 组件的 `include/defines.inc.php` 源程序文件中定义的；资源 ID 字段（`resourceid`），用于记录图表中某个单元格所放置资源的 ID 值；单元格宽度字段（`width`），用于记录图表中某个单元格的宽度，可取的值为正整数；单元格高度字段（`height`），用于记录图表中某个单元格的高度，可取的值为正整数；列字段（`x`），用于记录图表中某个资源被放置在图表中的列数；行字段（`y`），用于记录图表中某个资源被放置在图表中的行数；纵向对齐方式字段（`valign`），用于记录资源在单元格中纵向对齐方式，可取的值为 1、0、2，分别对应于“顶部”、“居中”和“底部”对齐方式；横向对齐方式字段（`halign`），用于记录资源在单元格中横向对齐方式，可取的值为 1、0、2，分别对应于“左侧”、“居中”和“右侧”对齐方式；是否是动态项目字段（`dynamic`），用于记录图表中某个单元格中被放置的资源项目是否属于动态项目，可取的值为 0 和 1，分别对应于不是动态项目和是动态项目。

3. 数据图表（graphs）

数据图表用于记录系统中所有数据图的信息，该表中包含以下字段信息：数据图 ID（`graphid`），它是数据图表的主键；数据图名（`name`），该字段的内容是不为空且长度不超过 128 个字符的字符串；宽度（`width`），该字段用于记录数据图被设定的宽度，它的允许值范围是小于等于 11 位的正整数（但是通常情况下，我们很少将数据图的宽度设置成超过 1000 个像素）；高度（`height`），该字段用于记录数据图被设定的高度，它的允许值范围与宽度字段一样，也是小于等于 11 位的正整数；数据图类型（`graphtype`），该字段用于指定数据图属于“一般数据图”

(用数字 0 表示)、“层叠图”(用数 1 表示)、“饼形图”(用数字 2 表示)还是“切开的饼形图”(用数字 3 表示); 标签(tags), 该字段用于表示数据图属于哪个级别的数据图, 它可取 0、2、4 之中的任一整数, 分别表示对应的数据图为用户在主机级别上所定义的普通数据图、数据图模板和系统依据数据图模板在主机上所创建的真实数据图。

4. 数据图项目关系表 (graphs_items)

我们知道, 在一张数据图中可以引用多个监控项目的数据, 该表即是用于记录数据图与监控项目之间关系的数据表, 它包括以下主要字段: 数据图关系 ID (gitemid), 该字段是数据图项目关系表的主键; 数据图 ID (graphid), 它是数据图表的外键; 项目 ID (itemid), 它是监控项目表的外键; 监控项目类型 (type), 即指定数据图所引用的监控项目是直接创建在主机上的普通监控项目还是项目样板, 分别用整数 0 和 2 表示。

5. 监控项目表 (items)

监控项目表中的字段非常多, 多达几十个, 但是我们的批量添加图表程序中只需使用该表中以下几个字段: 监控项目 ID (itemid), 它是监控项目表的主键; 主机 ID (hostid), 它是主机表的外键, 用于表示指定的监控项目属于哪个主机; 监控项目名称 (name), 该字段的作用比较好理解, 用于记录对应监控项目的名称, 它的值为非空且长度不大于 255 个字符的字符串; 信息类型 (value_type), 用于指定监控项目所采集的监控数据属于哪种信息类型, 其取值范围为 0~4 的整数, 分别代表“数值(浮点数)”、“字符串”、“日志”、“数值(无符号整型)”和“文本”型; 标签 (tags), 该字段用于标识监控项目属于哪个级别的监控项目, 它可取 0、1、2、4 之中的任一整数, 分别表示对应监控项目属于用户直接创建在主机上的普通监控项目、低级自动发现规则(注意, 不是指项目样板, 而是指低级自动发现规则本身。在 Zabbix 系统中, 一个低级自动发现规则与一个普通的监控项目在本质上是同样的)、项目样板和系统根据某个项目样板在主机上所创建的实际监控项目。

6. 主机表 (hosts)

与监控项目表类似, 主机表也有多达几十个字段, 但是我们只需使用该表中的以下几个字段: 主机 ID (hostid), 它是主机表的主键; 主机名 (host), 它的取值范围为非空且长度不大于 64 个字符的字符串, 用于记录主机的 IP 地址或主机的 DNS 主机名; 主机显示名称 (name), 该字段记录的是主机的显示名称, 其内容为不大于 64 个字符的非空字符串。

7. 入侵检测表(ids)

由这个数据表的名称不难看出, 这是一张用于防止 Zabbix 系统数据库被非法修改、增强 Zabbix 系统安全的数据表。在该表中记录了 Zabbix 系统中, 比较重要的数据表的表名以及它们当前最大的 ID 值。当有非法用户通过非法的手段修改了系统中某个表的记录, 且又没有对 ids 数据表进行相应的修改, 则之后合法用户通过 Web 组件前台页面进行正常操作时, 系统就可能报错。入侵检测表中的字段比较简单, 主要包含以下这么几个字段: 入侵检测表 ID (nodeid), 该字段是入侵检测表的主键; 表名 (table_name), 用于记录 Zabbix 系统数据库中数据表的名称, 例如 items、hosts 等; 字段名称 (field_name), 用于记录 Zabbix 系统数据库中数据表所对应的主键字段名称, 例如该表中某条记录的表名字段内容为 items, 则对应记录的该字段内容为监控项目表中主键字段名称, 即 itemid; 下一个 ID 值 (nextid), 笔者经实际测试发现该字段实际存储的值与它的名称所暗示的值还是有一点出入的, 例如, 假如系统中当前监控项目表中

监控项目 ID 最大值为 100，如果按照该字段名称的字面含义来理解，则可能很容易认为在入侵检测表中对应记录的 nextid 字段值为 101，但是经实际测试我们发现，该值并非为 101 而是 100。

上面对将要开发的批量添加图表程序中可能要使用到的数据表以及它们之间的关系和作用做了一个简单的说明和介绍。但是需要提醒注意的是，上述介绍的这些数据表，仅仅是将要开发的批量添加图表程序中可能会使用到的数据表，而绝非是整个 Zabbix 系统数据库中只有这些数据表，而且整个 Zabbix 系统数据库中各数据表之间的关系也远比图 9-9 所示的关系要复杂得多。

9.3.3 程序流程分析

完成上面所述的数据表之间的关系分析后，接下来，就要进行具体的程序开发了。但在编写批量添加图表程序之前，我们先来看看该程序的程序流程，如图 9-10 所示。

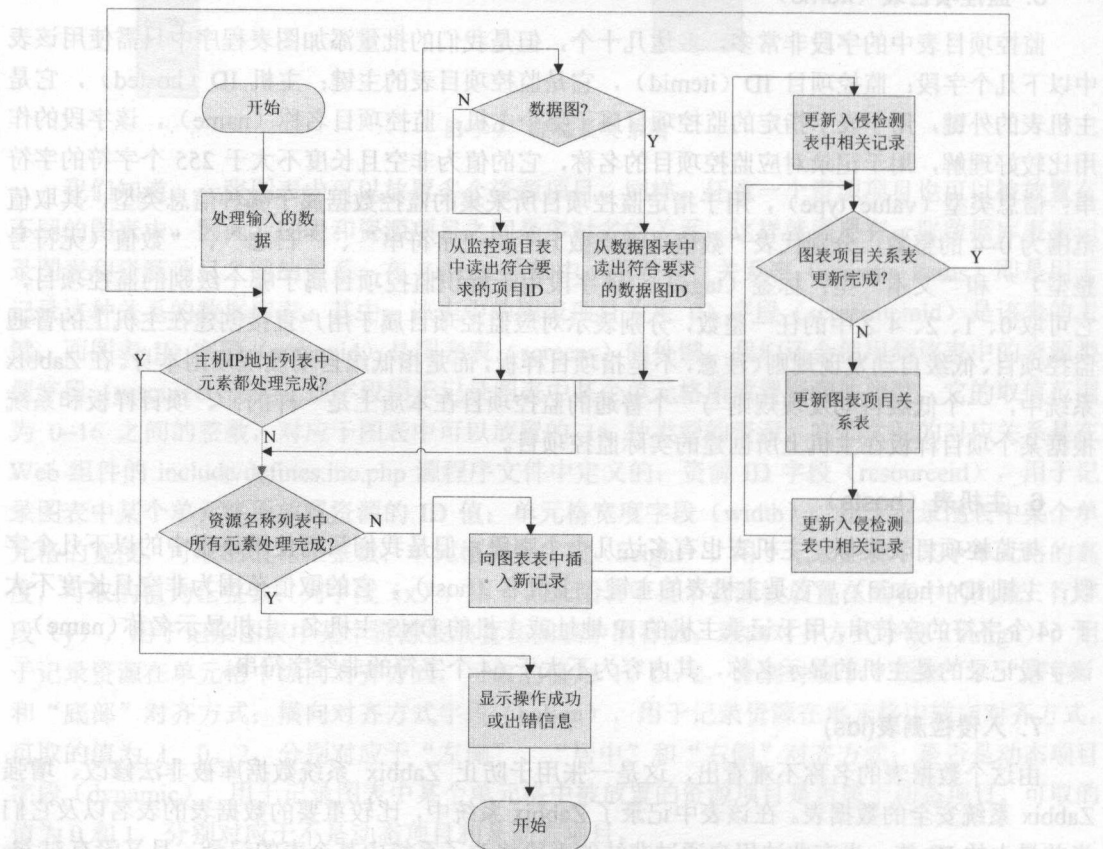


图 9-10 批量添加图表程序流程图

结合图 9-10 所示的程序流图和之前对数据表所作的分析以及所需要实现的功能，接下来就可以进行程序源代码的编写了。为了使所编写的页面与 Zabbix 系统 Web 组件的其他页面在风格和样式上保持一致，同时也为了减轻我们编写代码的工作量，充分利用系统中现有的源代码资源，这里以系统中已存在的 screenedit.php 程序文件为基础，在此基础上进行新程序的开发。故此，首先需要复制系统中的 screenedit.php 程序文件，并将新复制的文件重命名为 screensbatch.php，然后删除该文件中“check_fields(\$fields);”行到最后一次包含“require_once

`dirname(__FILE__).'/include/page_footer.php';`”内容所在行（注意，不删除该行）之间的所有内容，即删除内容后的 `screensbatch.php` 文件内容类似于图 9-11 所示。

```
<?php
/*
** Zabbix
** Copyright (C) 2000-2011 Zabbix SIA
.....(以下略去若干行内容) ①
**/
.....(以上略去若干行内容)
$fields = array(
    'screenid' =>    array(T_ZBX_INT, O_MAND, P_SYS,    DB_ID,                null),
    .....(以下略去若干行内容)
    'ajaxAction' =>    array(T_ZBX_STR, O_OPT, P_ACT,    null,                null)
);
(删除此处源代码，并将新编写的源代码书写在此处) ③
require_once dirname(__FILE__).'/include/page_footer.php';
?>
```

图 9-11 `screensbatch.php` 程序文件内容截图

由于篇幅限制，在图 9-11 所示的①和②区域都省略了若干行，而③区域的内容正是需要删除的内容，并且将要编写的源代码就将书写在此处。以下便是批量添加图表功能的程序源代码。

```
1. if(isset($_REQUEST['action'])) $action=trim($_REQUEST['action']);
2. else $action=0;
3. if($action != 1){
4. ?>
5. <div><table class="ui-widget-header ui-corner-all header maxwidth"
   cellspacing="0" cellpadding="1">
   <tr><td class="header_l left">批量添加图表</td><td class="header_r right"><div
   class="floatright">&nbsp;</div></td></tr></table></div>
   .....
6. <?php
7. }elseif(
8. $colnum=trim($_REQUEST['colnum']);
9. $resourcetype=trim($_REQUEST['resourcetype']);
10. $hostip=trim($_REQUEST['hostip']);
11. $resourcelist=trim($_REQUEST['resourcelist']);
12. $width=trim($_REQUEST['width']);
13. $height=trim($_REQUEST['height']);
14. $halign=trim($_REQUEST['halign']);
15. $valign=trim($_REQUEST['valign']);
16. if(!isset($_REQUEST['dynamic']) || trim($_REQUEST['dynamic']) == "")
   $dynamic=0;
17. else $dynamic=1;
18. $iparray=explode(",",$hostip);
19. $resource_array=explode(",",$resourcelist);
20. $screenid_array = DBfetch(DBselect('select max(screenid) as id from
   screens;'));
21. $screenid=$screenid_array['id'];
22. for($i=0;$i<count($iparray);$i++){
23.     $resourcecount=0;
24.     for($j=0;$j<count($resource_array);$j++){
25.         if($resourcetype == "0"){
26.             $query="select a.graphid as id from graphs a,graphs_items b,
```



```

hosts c ,items d ";
27.         $query .= "where a.name like '%" . $resource_array[$j] . "%' and
a.graphid=b.graphid and ";
28.         $query .= "b.itemid=d.itemid and c.hostid=d.hostid and
c.host='" . $iparray[$i] . "' and a.flags in(4,0) group by a.graphid;";
29.     }else{
30.         $query="select a.itemid as id from items a, hosts b where
a.hostid=b.hostid and b.host='" . $iparray[$i] . "' and a.name like '%" . $resource_
array[$j] . "%' ";
31.         $query .= "and a.flags in(4,0) and a.value_type in(0,3);";
32.     }
33.     $result=DBselect($query);
34.     while($row=DBfetch($result)){
35.         $resourceidarray[$iparray[$i]][$resourcecount]=$row['id'];
36.         $resourcecount++;
37.     }
38. }
39. DBexecute("insert into screens values('".($screenid+1)."',
'". $iparray[$i] . "',".
$colsnum." ,".ceil($resourcecount/$colsnum)." ,NULL);");
40. DBexecute("update ids set nextid='".($screenid+1)." where table_name=
'screens';");
41. $screenitemid_array = DBfetch(DBselect('select max(screenitemid) as sid
from screens_items;'));
42. $screenitemid=$screenitemid_array['sid'];
43. for($l=0;$l<$resourcecount;$l++){
44.     $query="insert into screens_items values('".($screenitemid+1)." ,".
($screenid+1)." ,". $resource_type ." ,". $resourceidarray[$iparray[$i]][$l];
45.     $query .= " ,". $width ." ,". $height ." ,". fmod($l,$colsnum)." ,". floor
($l/$colsnum)." ,0,0,0 ,". $valign ." ,". $halign ." ,0 ,' ,". $dynamic ." ,0);";
46.     DBexecute($query);
47.     DBexecute("update ids set nextid='".($screenitemid+1)." where
table_name='screens_items';");
48.     $screenitemid++;
49. }
50. $screenid++;
51. }
52. show_messages(true,"以列表". $hostip ."中各主机 IP 和主机名命名的图表添加成功");
53. }

```

同样是出于篇幅方面的考虑，上面所贴出的源代码中第 5 到第 6 行之间略去了若干行页面表单的 HTML 代码。所略去的这部分 HTML 代码不影响我们对整个程序逻辑的理解，而且这部分代码内容比较简单，它只是表单部分的代码，相信读者结合图 9-8 所示的表单页面截图，可以很容易地补齐这部分 HTML 代码。只是读者在补齐这部分被省略的 HTML 代码时，需要注意表单项和它的名称之间的对应关系为：图表列数->colsnum；图表资源类型->resourcetype；主机 IP 地址列表->hostip；资源名称列表->resourceidlist；宽->width；高->height；横向对齐->halign；纵向对齐->valign；动态项目->dynamic；最后还有一个隐藏的表单项，用于标识表单是否被提交，它的名称为 action，值为 1。

接下来，对上面所贴出的这段源代码作一个简单的说明。第 1 行 if 语句用于判断表单是否被提交，如果未被提交，则输出表单的 HTML 源代码，也即第 6 行之前的内容。从第 6 行到第

19 行的代码用于处理用户输入的数据，并将它们赋值给各自的变量。第 20、第 21 行查询系统中当前最大的 `screenid` 值，并将所查询的值赋给变量 `$screenid`。在第 20 行中，使用了 Zabbix 系统 Web 组件中自定义的两个数据库操作函数 `DBfetch` 和 `DBselect`。这两个函数都是在 Web 组件的 `include/db.inc.php` 程序文件中定义的。其中函数 `DBfetch` 的功能和作用与 PHP 语言中 `mysql_fetch_assoc` 函数的功能和作用是类似的，即从数据库查询的结果集中取出一行生成关联数组，并返回该关联数组；而 `DBselect` 函数的功能和作用与 PHP 语言中 `mysql_query` 函数的功能和作用是类似的，即执行指定的 SQL 语句查询，并返回所查询的结果集。

使用 Zabbix 系统 Web 组件自定义的数据库查询函数不但可以减少程序开发的工作量，而且可以使新开发的程序能够更好地融入到 Zabbix 系统 Web 组件中，使其和 Zabbix 系统 Web 组件看起来更像是一个完整的整体。同时，我们知道 Zabbix 系统是支持包括 MySQL、Oracle 在内的众多关系数据库的，而 PHP 语言对于不同的关系数据库有着不同的接口函数，使用 Zabbix 系统 Web 组件自定义的数据库操作函数，可以使我们在编写程序时无须考虑这些数据库接口函数不同的问题，从而使我们开发出来的程序“自然而然”地支持多种关系型数据库。由此，笔者建议在对 Zabbix 系统进行二次开发时，要尽可能使用 Zabbix 系统中原先就定义好的自定义函数、变量等，这样不但可以减少开发新程序的工作量，而且还可以使新开发的程序与 Zabbix 系统原有的程序组成一个完整的整体，不至于被割裂开来。

从第 22 行开始到第 51 行结束是一个大的 `for` 循环，它的循环次数是由用户在主机 IP 地址列表表单中所输入的元素个数决定的。每完整循环一次，程序就会以 IP 地址列表中指定元素为名称在系统中创建一个图表，并在系统中搜索指定类型资源名称中包含指定字符串的资源，同时将所搜索到的资源作为新创建图表的资源，放置在新创建图表的指定单元格中。其中，第 24 行到第 38 行又是一个 `for` 循环体，即第二重循环体，它针对用户所输入的资源名称列表中每个元素，在系统数据库中搜索需要的资源，并将搜索到的资源 ID 记录到 `$resourceidarray` 数组中。在这个 `for` 循环体中，每循环一次，程序都执行一次 SQL 查询，而所执行的查询语句是分 `$resourcetype` 变量值等于 0（即用户在图表资源类型表单项中选择了“数据图”项）和不等于 0（即用户在图表资源类型表单项中选择了“简单数据图”项）的。对于这里所执行的两类查询语句，我们结合前面对数据表关系所作的分析应该不难理解，只是这两类查询语句中的“`a.flags in(4,0)`”和“`value_type in(0,3)`”可能需要稍微说明一下。我们知道，在图表中，标签（tags）字段用于标识数据图属于哪个级别的数据图，它可取 0、2、4 之中的任一整数，分别表示对应的数据图为用户在主机上所定义的普通数据图、数据图模板和系统依据数据图模板在主机上所创建的真实数据图。因为本质上可以将数据图模板视为一种规则，它没有对应的监控数据，所以不是真正意义上的“图”，也就不能放置在图表中。这样，在我们查询准备放置在图表上的资源时，就需要将这类数据图给排除掉，所以在查询语句中 `flags` 字段的值就只能是 4 或 0。类似的，在监控项目表中，`flags` 字段值为 0 或 4 的记录所代表的监控项目为普通监控项目和系统依据项目样板而创建的真实项目，只有这两类监控项目才有监控数据，也才可能有简单数据图。所以，在查询语句中 `flags` 字段的值就只能是 4 或 0。而字段 `value_type` 则表示的是监控项目的信息类型。我们知道，只有信息类型为“数值（浮点数）”和“数值（无符号整型）”的监控项目才会有简单数据图，而这两种类型的监控项目在监控项目表中对应 `value_type` 字段的值分别为 0 和 4，所以在查询语句中需要将 `value_type` 字段的值限定为 0 和 4。

接下来的第 39 行和第 40 行，则分别是向图表表中插入新的记录和更新入侵检测表中相关记录。在这两行中，使用了 Zabbix 系统 Web 组件中另一个自定义函数——`DBexecute`，该函数

也是在 Web 组件的 `include/db.inc.php` 程序文件中定义的，它的功能和作用类似于 PHP 语言中 `mysql_query` 数据库操作函数，即执行对指定 SQL 语句的查询。

而第 43 行到第 49 行又是一个 for 循环体，该循环体的作用是用之前所查询到的资源信息，更新新建图表的资源，即向图表项目关系表中插入相关的新记录，并更新入侵检测表中相关记录的数据。最后的第 52 行是显示操作成功的提示信息。

这里所提供的批量添加图表程序，主要用于说明如何针对 Zabbix 系统 Web 组件进行二次开发。因此，虽然这个程序在功能上是完整的，而且也是可以正常使用的，但是仍然有需要优化的地方。比如，在该程序中没有对用户输入的数据进行有效性校验，也没有对异常错误进行处理，而只是在程序的最后给出了操作成功的提示。对于这些可以优化和改善的地方，有兴趣的读者可以自行研究并对该程序进行必要的优化。

9.4 如何添加自定义菜单项

当针对 Zabbix 系统 Web 组件做了某些新功能的开发，并完成了相应程序文件的编写，且调试无误后，接下来我们当然希望针对新开发的功能页面在系统中增加相应的新菜单项，以使新开发的功能能够作为 Web 组件的一部分，完美地整合到整个 Web 组件中去。其实添加或修改 Zabbix 系统菜单项并不复杂，这里就以将批量添加图表功能页面添加到 Zabbix 系统菜单中为例，简单介绍一下如何添加和修改 Zabbix 系统菜单项。

9.4.1 添加和修改菜单项

Zabbix 系统的菜单项是在其 Web 组件的 `include/menu.inc.php` 程序文件中定义的，以下是从该文件中截取的一段定义系统菜单项的代码，让我们先来看看这段代码。

```

/*****include/menu.inc.php 文件中部分内容 *****/
.....
1. $ZBX_MENU = array(
2.     'view' => array(
3.         'label' => _('Monitoring'),
4.         'user_type' => USER_TYPE_ZABBIX_USER,
5.         'node_perm' => PERM_READ_LIST,
6.         'default_page_id' => 0,
7.         'pages' => array(
8.             array(
9.                 'url' => 'dashboard.php',
10.                'label' => _('Dashboard'),
11.                'sub_pages' => array('dashconf.php')
12.            ),
13.            array(
14.                'url' => 'overview.php',
15.                'label' => _('Overview')
16.            ),
17.            array(
18.                'url' => 'httpmon.php',
19.                'label' => _('Web'),
20.                'sub_pages' => array('httpdetails.php')
21.            ),

```



```

22.         array(
23.             'url' => 'latest.php',
24.             'label' => _('Latest data'),
25.             'sub_pages' => array('history.php', 'chart.php')
26.         ),
27.         ...
28.         array(
29.             'url' => 'jsrpc.php'
30.         )
31.     ),
32.     ...

```

由上面这段源代码不难看出，Zabbix 系统的菜单项信息是定义在一个叫做 \$ZBX_MENU 的多维数组里的。而这个多维数组中的每个第一维（也就是第一级）元素，就定义了一个 Zabbix 系统主菜单项及其子菜单项的信息。例如，上面的这段代码中键为“view”的元素，就定义了 Zabbix 系统中“状态统计”及其下子菜单项的信息。然而，这里需要提醒注意的是，虽然 Zabbix 官方提供的源程序中，所定义的 \$ZBX_MENU 数组中第一维元素的个数有 6 个，这其中包括定义“登录(Login)”主菜单项及其子菜单项所对应的第一维元素，但是通常情况下我们通过 Web 组件页面只能看到 5 个主菜单项。这是因为，“登录(Login)”主菜单项在通常情况下是隐藏不显示的。为什么在通常情况下不显示“登录”主菜单项呢？其实道理也比较简单，出于安全方面考虑，如果尚未登录入系统，Zabbix 系统是不允许你进行除了登录之外的其他任何操作的，此时 Zabbix 系统只显示一个登录表单，以供你登录入系统使用。既然在用户登录入系统之前，系统不允许用户执行除登录之外的其他任何操作，则此时系统就无须在登录表单页面上显示任何菜单项，包括“登录”主菜单项。而当用户成功登录入系统后，“登录”这个功能页面对用户来说就不再有任何作用了。因此，此时系统同样也没有必要显示“登录”这个主菜单项。既然在用户尚未登录入系统之前，系统不需要显示“登录”主菜单项，而在用户登录入系统之后同样也没有必要显示“登录”主菜单项，那么这个“登录”主菜单项当然就可以被隐藏掉。既然“登录”主菜单项在系统中是一直被隐藏的，那么为什么还需要定义这么一个主菜单项呢？这是因为，虽然“登录”这个主菜单项被系统隐藏掉了，但是登录仍然是用户必须执行的一个操作步骤，系统需要对这类操作行为记录审计日志。因此，还必须在系统中定义它。

\$ZBX_MENU 数组的第二维元素定义的是系统主菜单项的一些属性以及对应主菜单项下的子菜单项信息。其中，键 label 用于指定主菜单项的名称。因为，在源程序中所定义的主菜单项名称，将来可能需要被翻译成非英语语言，因此主菜单项的名称需要用标识符标识出来，即书写成_('Dashboard')形式。而键 user_type 则用于指定，哪些类型的系统用户有权限访问该主菜单项及其子菜单项。我们知道，在 Zabbix 系统中，除了来宾用户(guest 用户)以外，还有另外三种类型的用户，它们是：“Zabbix 用户”、“Zabbix 管理员”和“Zabbix 超级管理员”。对应于这三种用户类型，键值为 user_type 的元素，其应取的值为 USER_TYPE_ZABBIX_USER、USER_TYPE_ZABBIX_ADMIN 和 USER_TYPE_SUPER_ADMIN。而这三种类型用户的权限是依次递增的，因此如果将定义某个主菜单项 user_type 键所对应元素的值设置成 USER_TYPE_ZABBIX_USER，则该主菜单项及其下子菜单项可以被除来宾用户以外的其他所有类型用户访问到。反之，如果将其值设置为 USER_TYPE_SUPER_ADMIN，则就只有“Zabbix 超级管理员”用户才可以访问该主菜单项及其下的所有子菜单项。而如果键 user_type 所对应元素的值被设置成 0，则表示所有用户，包括来宾用户，都可以访问该主菜单项及其下的所有子菜单项。需

要注意的是，在 Zabbix 系统中，我们上述所讨论的权限是针对功能页面或者说菜单项而言的。它的具体体现是，当不具有某一权限的用户使用 Web 组件页面时，他将看不到对应的主菜单项，也就没有办法访问对应的功能页面。但是，如果你认为 Zabbix 系统只是靠隐藏菜单项这个小伎俩，来实现系统权限控制的话，那你也太小看 Zabbix 系统了。实际上，当你不具有访问系统中某个菜单项的权限时，即使不通过菜单系统操作，而直接在浏览器的地址栏中，输入功能页面的 URL 地址，强行访问也是被禁止的。在 Zabbix 系统中，除了上述的这种基于功能页面的权限控制体系以外，还有另外一套权限控制体系，那就是针对系统中所配置的被监控元素的，即基于用户组所配置的权限控制体系。关于这套权限控制体系的管理和配置，在前面相关章节中已经作过非常详细的介绍和说明，在此就不再一一地复述。

接下来的键 `node_perm` 所对应的元素，用于指定用户对分布式节点相应功能页面，是否拥有访问权限。这个元素的值可以取 `PERM_DENY`、`PERM_READ_LIST`、`PERM_READ_ONLY`、`PERM_READ_WRITE` 和 `PERM_MAX` 中的任何一个，相对应的权限依次递增。但是，当这个数组元素的值被分别设置为 `PERM_READ_WRITE` 和 `PERM_MAX` 时，对于用户来说，具有相同的权限。因为在 Zabbix 系统中，同时具有“读”和“写”的权限，实际上就是系统中的最大权限了。实际上，上面所述的这些数组元素可以被设置的值，在 Zabbix 系统的源代码里，其实就被定义成了各自对应的常量。而常量 `PERM_READ_WRITE` 和 `PERM_MAX` 的值，在源代码里都被设置成 3。

定义菜单信息数组中的“`default_page_id`”元素，用于指定主菜单项的默认页面。不难理解，任何一个主菜单项本身是不需要有其自己的功能页面的，因为任何一个主菜单项的功能，都是通过其下的子菜单项来完成的。但是，当用户单击主菜单项时，系统总是需要显示某些信息的。因此，系统就需要为每个主菜单项指定一个默认显示的页面，而这个默认被显示的页面就是通过 `default_page_id` 元素来指定的。该元素的取值范围为 0 到小于对应主菜单项下子菜单项个数之间的整数。例如，“状态统计”主菜单项下有 11 个子菜单项，那么，该元素的取值范围就是 0~10 之间的整数，而它的含义为对应主菜单项下第几个子菜单项所对应的页面。例如，默认情况下，定义“状态统计”主菜单项数组中的 `default_page_id` 元素的值被设置为 0，则“状态统计”主菜单项的默认页面即为它的第一个子菜单项的功能页面——“状态面板”页面。

通过上面给出的源代码，不难看出，`$ZBX_MENU` 数组中键值为 `pages` 的元素，用于配置主菜单项下子菜单项的信息。而该元素的值还是一个数组，即通过它定义了 `$ZBX_MENU` 这个多维数组中的第三维和第四维元素。其中，每个第三维元素都对应一个子菜单项，而第四维元素用于配置子菜单项的具体信息，包括键值为 `url` 的元素，用于配置子菜单项对应功能页面真实的 URL 地址。与主菜单项类似，键值为 `label` 的元素，则用于指定子菜单项的名称，如果该元素的值为空或者没有配置该元素，则对应的子菜单项将不会在 Web 组件的页面上显示出来。不过，通过其他入口，浏览过该子菜单项的功能页面后，系统就会在“历史数据”栏中显示相应的记录。我们知道，在 Web 组件的某些页面中，可能会引用到其他功能页面。例如，当在“最新数据”页面上单击“数据图”超链接时，就可以查看对应监控项目的简单数据图，而我们知道简单数据图的功能页面与“最新数据”的功能页面并不是同一个功能页面。像这种被某个功能页面引用的功能页面的 URL 地址，就需要被配置在键值为 `sub_pages` 的元素中，当有多个这样的功能页面的 URL 地址需要配置时，就用半角的逗号分隔（,）它们。

上面对定义 Zabbix 系统菜单的 `$ZBX_MENU` 多维数组中每类元素的功能和含义作了简单的说明，接下来就需要编写具体的代码，将批量添加图表功能页面添加到 Zabbix 系统菜单中。

不妨将要添加的主菜单项叫做“管理工具 (Admintools)”，而它的默认页面为我们之前所开发的批量添加图表程序所对应的功能页面。为了有更好的显示效果，我们不妨在这个主菜单项下面配置两个子菜单项，分别叫做“批量添加图表 (Screensbatch)”和“测试子菜单 (Testmenu)”，对应的功能页面分别为 `screensbatch.php` 和 `testmenu.php`。而添加的菜单项所对应的功能为批量添加图表，这是一个对 Zabbix 系统有较大更改的操作，所以需要将这个菜单项的权限设置为只有 Zabbix 超级管理员用户才能执行。这样，我们需要在 `$ZBX_MENU` 多维数组中添加如下所示的代码，只需将下面这段代码添加到 `$ZBX_MENU` 数组的 `admin` 元素后面即可。

```

1. 'admintools' => array(
2.     'label' => _('Admintools'),
3.     'user_type' => USER_TYPE_SUPER_ADMIN,
4.     'node_perm' => PERM_READ_WRITE,
5.     'default_page_id' => 0,
6.     'pages' => array(
7.         array(
8.             'label' => _('Screensbatch'),
9.             'url' => 'screensbatch.php',
10.        ),
11.        array(
12.            'label' => _('Testmenu'),
13.            'url' => 'testmenu.php',
14.        )
15.    )
16. ),

```

虽然，将上面这段代码添加到 `include/menu.inc.php` 文件中之后，就可以从 Web 组件的前台页面上，看到我们所添加的主菜单项以及它的两个子菜单项了，但是，此时如果将鼠标从新添加的菜单项上移到别的菜单项上，则系统并不会像我们操作其他菜单项一样，隐藏掉新添加的主菜单项下的子菜单项。因此，为了添加鼠标移动效果，还需要修改 `js/main.js` 文件。在该文件中“`menus: { 'empty': 0, 'view': 0, 'cm': 0, …… }`”内容所在的行，添加上新添加的主菜单项名称，如图 9-12 所示的红色方框所标识的内容。



```

admin: {
  menus: {
    def_label: null,
    { 'empty': 0, 'view': 0, 'cm': 0, 'reports': 0, 'configs': 0, 'admin': 0, 'admintools': 0, 'login': 0,

```

图 9-12 `js/main.js` 文件中需要修改的内容

9.4.2 汉化菜单项

当按照上面介绍的方法添加完主菜单项及其子菜单项后，通过 Web 组件的前台页面就可以看到所添加的菜单项效果了。但是很快你就会发现，我们所添加的菜单项，在页面上是以英文形式显示的。这是因为，我们还没有对新添加的菜单项名称进行汉化，接下来我们对这几个菜单项名称进行汉化。

如果你对前面介绍的如何完善 Zabbix 系统的汉化效果的内容还有一些印象，则你或许已经注意到了，在我们添加新的菜单项时，实际上已经在 Zabbix 系统 Web 组件的源程序中增加了新的信息 ID。因此，按照前面所介绍的方法，在具体汉化菜单项名称之前，需要更新可移植对象文件内容，其方法是执行下列命令：

```

#切换到 Web 组件部署目录的 local 子目录下
shell> cd /data/website/zabbix/locale

```



```
#执行 update_po.sh 脚本程序，以更新可移植对象文件内容
shell> ./update_po.sh
```

提示：如果在执行 update_po.sh 脚本程序时报“keyword msgctxt unknown”错误，则很可能是因为你的系统中 gettext 软件包的版本太低引起的，这时需要将 gettext 软件包升级到 gettext-0.17 以上，再执行该脚本程序。

完成可移植对象文件内容的更新后，就可以用任何一款编辑器，打开 Web 组件部署目录下的 locale/zh_CN/LC_MESSAGES/frontend.po 文件，并将信息 ID 为 Admintools、Screensbatch 和 Testmenu 等的信息依次翻译成管理工具、批量添加图表和测试子菜单。

翻译好各菜单项的名称后，还需要执行 Web 组件部署目录下的 locale/make_mo.sh 脚本程序，以更新系统中的机器对象文件内容。

最后，为了使新添加的菜单项能够在被选中时显示被选中的效果，并且使在打开新添加的批量添加图表功能页面时，该页面的标题能显示成自己的“管理工具”页面标题，还需要对 screensbatch.php 程序页面稍加修改：将“\$page['title'] = _('Configuration of screens');”行的内容修改成“\$page['title'] = _('Admintools');”；将“\$page['file'] = 'screenedit.php';”行内容修改为“\$page['file'] = 'screensbatch.php';”。

好了，完成上面这些修改后，用浏览器打开 Web 组件的前台页面，就可以看到如图 9-13 所示的效果了。

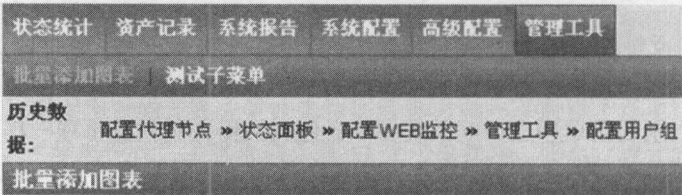


图 9-13 新添加菜单项效果图

9.5 为何数据图经常出现断图

在日常维护 Zabbix 系统的过程中，数据图（包括简单数据图）断图可能是我们所遇到的最常见问题。在前面章节中，关于 Zabbix 系统的数据图断图问题及其产生的原因，也陆续作过一些讨论。本节我们将对数据图断图产生的原因及其解决方法等进行较详细的讨论。

9.5.1 数据图断图根本原因分析

根据我们经验，其实不仅仅是 Zabbix 系统，几乎所有的监控系统在某些情况下都会出现断图的情况。那么造成数据图断图的根本原因都有哪些呢？毫无疑问，我们只有找出造成数据图断图的根本原因，才有可能找到解决数据图断图问题的根本方法。首先，让我们来看看图 9-14 所示的数据图。

由图 9-14 可以看出，监控服务器的系统负载值数据图在 10:23 到 10:28 之间有一次中断。我们知道，Zabbix 系统中的数据图是 Web 组件中相关的 PHP 程序通过读取数据库中的数据，使用 GD 库实时绘制的。因此，如果数据图出现问题，无非是由这么两个原因造成的：第一，

绘制数据图的 PHP 程序执行出错；第二，绘制数据图的 PHP 程序没有成功地从数据库中读取到绘制数据图所需要的相关数据。

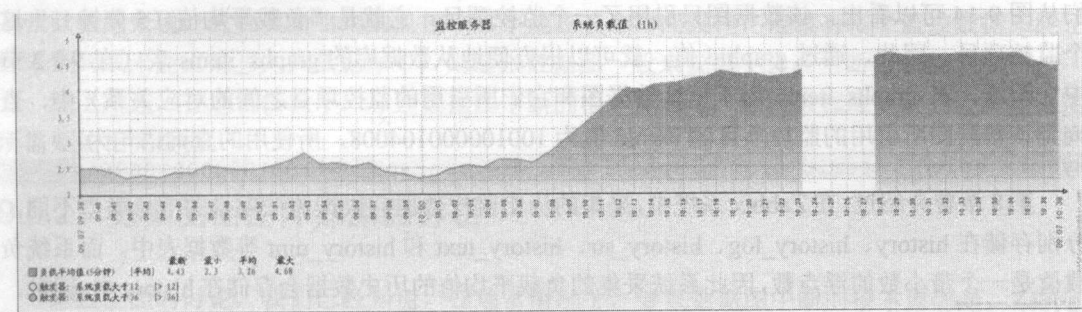


图 9-14 监控服务器系统负载值数据图

使用任何一门高级编程语言基于 GD 库，开发过图形绘制程序的读者应该都有这样一个经验：如果执行绘图任务的程序运行出错，则目标图形将无法绘制出来。而图 9-15 所示的数据图，实际上程序绘制它是完全成功的，只是该图的一些部分没有达到我们期待的效果。因此，我们基本上可以断定，图 9-14 所示的数据图之所以会出现中断，并不是因为负责绘制数据图的相关 PHP 程序执行不成功造成的。而关于这一点，从 PHP 进程和 Web 服务器所产生的日志中，我们并没有找到疑似相关错误信息也可以得到印证。

既然排除了因为负责绘制数据图的 PHP 程序运行出错，而导致的数据图中断这一可疑点，那么，接下来的可能原因就只能是，因为 PHP 程序并没有完全读取到绘制数据图所需要的数据，从而导致数据图有中断。而对于部署正确、运行正常的 Zabbix 系统来说，如果负责绘制数据图的 PHP 程序，不能完整地查询到数据库中的完整数据，则不外乎有这么三个原因：第一，程序查询数据库时出现问题，比如查询数据库超时所导致的查询失败；第二，程序中查询语句的 Bug 导致无法查询到完整的数据；第三，数据库中压根就没有某些数据，很显然也就无从谈起查询到相应数据。

假如是第一种原因导致数据图中断，一般来说不管是何种具体原因导致 PHP 程序查询数据库失败，都会在系统中的 Web 前端页面或 Web 服务器日志文件中找到有关的错误信息，且与 PHP 程序运行不正常一样，这种情况下是根本无法绘制数据图的，而不仅仅是出现数据图中断。并且，如果仅仅因为超时导致数据库查询失败或查询到的数据不完整，且系统能够依据所查询到的部分数据绘制出不完整的数据图，那么，如果不停地刷新特定页面，则未能查询到的数据范围将是变化的，体现在数据图上的中断时间区间也将是变化的。因此，对于这种原因导致的数据图中断是非常容易识别和判定的。

对于第二种可能导致 Zabbix 系统无法完整地查询到所需数据的原因，显然已经超出本书的讨论范围，而且在实际维护 Zabbix 系统的过程中，极少发现有因为程序中的查询语句 Bug 而导致数据图中断的情况。

既然不是因为程序查询数据库失败而导致数据图中断，而又不大可能是因为程序中的查询语句 Bug 造成数据图中断，那么就只有一种可能原因会造成数据图中断，那就是数据库中缺少某些绘制数据图必需的数据，从而导致了数据图中断。下面就来实际验证一下造成数据图中断的根本原因，是不是因为数据库中缺少某些数据造成的。

图 9-14 所示的监控服务器系统负载值数据图在数据库中所对应的 graphid 值为 100100000016636，这个值在我们通过 Web 组件前台页面打开对应的数据图后（依次选择菜单项“状态统计”→“数

据图”，并在下拉菜单中选中需要查看的数据图），在数据图上单击鼠标右键，并在弹出的菜单中选择“属性”菜单项，这样我们在“属性”窗口的“地址”栏里就可以查看到该值。而我们从图 9-14 可以看出，该数据图只引用了一个监控项目，它就是“负载平均值（5 分钟）”这个监控项目。因此，通过 graphid 值，就可以很方便地从数据库的 graphs_items 表（在 9.3.2 节中介绍过，表 graphs_items 用于记录数据图和它们所引用的监控项目之间的对应关系）中，查询到该数据图所引用的监控项目的 itemid 值为 100100000104008，所使用的查询语句为：

```
mysql> select itemid from graphs_items where graphid='100100000016636';
```

第 8 章曾介绍过，在 Zabbix 系统中，采集的历史监控数据会根据监控项目的信息类型不同，分别存储在 history、history_log、history_str、history_text 和 history_uint 等数据表中。而系统负载值是一个带小数的浮点数，因此系统采集的负载平均值的历史数据会存储在 history 中。同时，系统中“负载平均值（5 分钟）”这个监控项目的监控数据采集间隔被设置成 60 秒。这样，按道理来说，在 history 数据表中，每分钟都会有一条 itemid 值为 100100000104008 的数据记录，因此下面来实际验证一下，看看情况是不是这样的：

```
mysql> select from_unixtime(clock,'%Y%m%d%H:%i:%S') as clock,value,itemid from
history where itemid='100100000104008' and clock>'1404872100' and
clock<'1404873300' order by clock;
```

上面这条查询语句的作用是：从 history 数据表中查询 itemid 字段值为 100100000104008，且 clock 字段值在 1404872100（2014 年 7 月 9 日 10 时 15 分 0 秒）到 1404873300（2014 年 7 月 9 日 10 时 35 分 0 秒）之间的记录，并将查询结果按 clock 字段值从小到大升序排列。执行上述查询后，系统将返回如图 9-15 所示的数据列表。

clock	value	itemid
20140709 10:15:32	4.1200	100100000104008
20140709 10:16:39	4.2000	100100000104008
20140709 10:17:22	4.2300	100100000104008
20140709 10:18:19	4.3500	100100000104008
20140709 10:19:25	4.2900	100100000104008
20140709 10:20:28	4.2800	100100000104008
20140709 10:21:20	4.2300	100100000104008
20140709 10:22:35	4.2700	100100000104008
20140709 10:23:17	4.3700	100100000104008
20140709 10:27:44	4.6200	100100000104008
20140709 10:28:36	4.6800	100100000104008
20140709 10:29:37	4.6300	100100000104008
20140709 10:31:27	4.6500	100100000104008
20140709 10:33:45	4.6700	100100000104008

图 9-15 监控服务器负载平均值（5 分钟）部分历史数据

由图 9-15 可以很容易看出：在 2014 年 7 月 9 日 10 时 23 分 17 秒时系统采集到一个监控数据后，大约有 4 分多钟的时间里，也就是到 10 时 27 分 44 秒之前，系统都没有采集到“负载平均值（5 分钟）”这个监控项目的任何一个监控数据。由此可见，图 9-15 所示的数据图之所以会有中断，其根本原因与之前的推测，即数据库中没有相应的数据记录是相一致的。

细心的读者可能已经发现了，在图 9-15 所示的数据列表中，除了在 10 时 23 分 17 秒到 10 时 27 分 44 秒之间缺失数条应有的数据记录外，在 10 时 29 分 37 秒到 10 时 31 分 27 秒之间以及 10 时 31 分 27 秒到 10 时 33 分 45 秒之间似乎也各缺失了一条数据。既然在这两个时间点也缺失了数据，那么为什么在图 9-14 所示的数据图中没有体现出来呢？其实这其中的缘由还是比较好理解的，这是因为，虽然针对监控项目设置的数据采集间隔是 1 分钟，但是系统基本上无法保证相邻的两次监控数据采集的时间间隔每次都刚好是 60 秒。同时，也不难理解，系统在绘制数据图时，会根据相邻的两个时间点所采集到的监控数据，估算这两个时间点之间其他时

刻的数据。否则,即使系统在每个计划时间点都能正确地采集到数据,并绘制这些数据所对应的图形,那么整张数据图看起来将会是一个个断断续续的点,而不是连续的图形。所以,对于某张数据图来说,只要其所引用的监控项目不是在某个时间段内,连续有多个监控数据未能正确采集到或未能正确写入到数据库中,则这张数据图一般来说都不会产生断图现象。

综上所述,在通常情况下,导致数据图中断的根本原因是,系统数据库中缺失绘制数据图所需要的相关数据。

9.5.2 数据图断图外部原因分析

通过上面的讨论,我们已经知道,在通常情况下,导致数据图中断的根本原因是,Zabbix 系统数据库中,因为某种原因导致绘制数据图所需要的相关数据缺失。那么,都有哪些外部原因,会导致在 Zabbix 系统数据库中缺失绘制数据图所需要的相关数据呢?因为每套实际运行中的 Zabbix 系统所基于的软件环境、硬件环境、网络环境都不尽相同,而且每台 Zabbix 服务器所监控的设备类型、数量以及所监控的项目也都不一样。因此,导致 Zabbix 系统数据库中,数据记录缺失的原因就非常多。这里我们无法列出所有的可能导致 Zabbix 系统数据库中数据缺失的外部原因,但是,依据我们日常维护的经验,以下这些因素可能是引起 Zabbix 系统数据库中数据缺失的最常见原因。

1. 管家进程引起的数据缺失

我们知道,在 Zabbix 系统中有一类进程叫做管家(housekeeper)进程。它的主要作用是,根据配置定期删除 Zabbix 系统数据库中过期的数据,以保证数据库中的记录不至于无限地膨胀。虽然管家进程也和 Zabbix 服务器端组件中其他进程一样,随着启动 Zabbix 服务器组件一起启动,并常驻在系统内存中。但是我们知道,管家进程并不是每时每刻都在删除数据库中过期的数据,而是根据在 `zabbix_server.conf` 配置文件中 `HousekeepingFrequency` 配置项的配置,每隔一定的时间集中删除数据库中过期的数据。既然管家进程是每隔一定的时间间隔,集中删除数据库中过期的数据,那么,如果将这个时间间隔设置得比较大,而系统中配置的监控项目数又比较多,同时监控数据采集的时间间隔又配置得比较小,则管家进程每次需要删除的数据量就比较大,对系统性能的占用也非常大,特别是对数据库性能的压力会变得非常大。在这种情况下,极易造成 Zabbix 服务器的历史数据同步器进程,无法及时地将系统所采集到的监控数据写入到系统数据库中的历史表中。或者轮询器进程无法及时地从被监控主机上采集到所需要的监控数据,从而导致相应时间段的历史数据丢失。以下这几张数据图就可以很好地印证这一点。

由图 9-16 可以比较容易地判断出:这台监控服务器上的管家进程是从 10 时 08 分开始执行数据清理任务的,因为从这个时刻开始管家进程的繁忙程度一下子就从 0% 上升到 100%(图 9-16 中①所示的实线),而到 10 时 47 分数据清理结束。同时,从图 9-16 还可以看出,由于管家进程的影响导致历史数据同步器进程(图中③所示的实线)和轮询器进程(图中②所示的实线)的繁忙程度也在 10 时 08 分开始急骤升高。这是因为,从图 9-16 中可以看出,这两种进程在正常情况下,其繁忙程度值是相对平稳的,而当管家进程繁忙程度升高时,这两种进程的繁忙程度也跟着急骤升高。因此,基本上可以断定历史数据同步器进程和轮询器进程繁忙程度的升高正是由于管家进程影响的。关于这一点,从图 9-17 所显示的 `mysql` 进程占用系统 CPU 的百分比也可以得到印证。

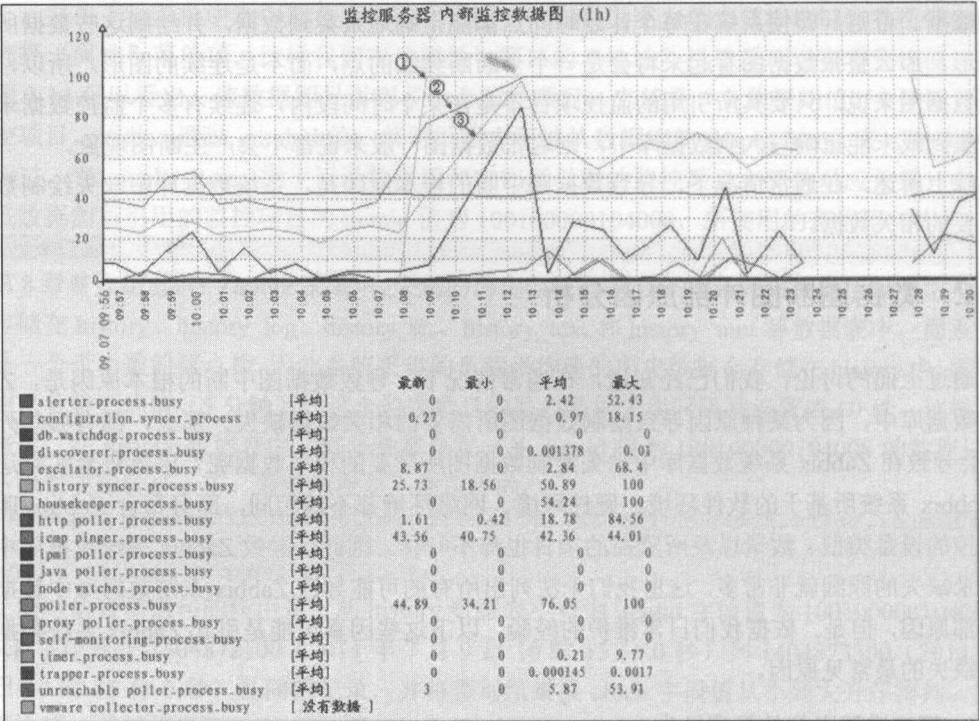


图 9-16 监控服务器内部监控数据图

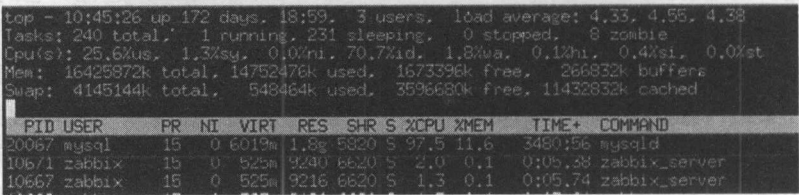


图 9-17 系统 top 命令数据截图

我们知道，管家进程的主要任务是删除数据库中过期的数据。因此，如果是由于管家进程的运行而导致系统在某些时间段丢失监控数据的话，则在数据库进程占用系统资源方面也应该有所体现。从图 9-17 可以看出，mysqld 进程占用 97.5% 的 CPU 资源。可见，此时系统的性能瓶颈是在 MySQL 数据库这一块，而这应该主要是由于管家进程大量操作系统数据库所引起的。

再者，由图 9-14 所示的监控服务器负载值数据图也可以看出，系统的负载值平时基本稳定在 2.5 左右（该服务器配置的是一个拥有 4 核 2.0G 至强 E5504 的 CPU），而当管家进程开始执行清除数据任务时，系统负载值就急骤上升并达到最大值 4.68。由此可以看出，正是由于管家进程对数据库的大量操作，才导致监控服务器遇到了硬件性能瓶颈。不仅如此，从图 9-18 所示的数据图还可以很容易地发现，在管家进程执行清除数据任务时，系统中产生了大量的数据采集延时的监控项目，而这毫无疑问更加剧了系统缺失历史数据的程度。

综上所述，在 Zabbix 系统中，管家进程是造成数据图中断的一种比较常见的原因，特别是在服务器硬件性能相对于所需要监控的对象数量不十分充裕的情况下，管家进程更容易造成数据图的中断。解决这个问题的最好方法是，禁用管家进程，而改用数据库的存储过程来管理 Zabbix 系统的历史数据。

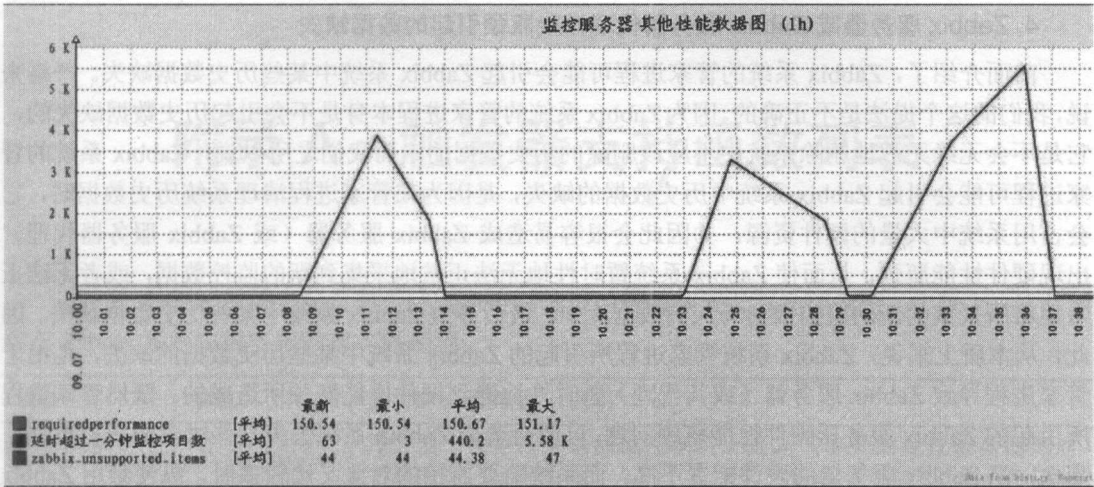


图 9-18 监控服务器其他性能数据图

2. 配置不当引起的数据缺失

在日常管理和维护 Zabbix 系统的过程中，因为维护人员的操作不当或对主机、监控项目或正则表达式配置错误也可能会造成数据缺失。例如，维护人员在修改主机配置时，将主机 IP 地址或者主机名配置错误，这将会导致该主机上所有的监控项目都无法正常地采集到监控数据。又例如，用户将监控项目的信息类型配置错误，比如系统实际采集到的数据是浮点型数据，而监控项目的信息类型被配置成了无符号整型，此时，系统在向数据库中插入新采集到的监控数据时就会报形如 “Received value [3.15] is not suitable for value type [Numeric (unsigned)] and data type [Decimal]” 的错误信息，该错误信息在 Web 组件的监控项目列表页面、Zabbix 服务器端组件的日志文件中和系统数据库的 items 表的 errors 字段中都能查看得到。最后，如果用户修改了某个正则表达式的规则，而使原本通过低级自动发现功能所发现的某些监控项目被错误过滤掉了，这样对应的监控项目，系统也就不会再采集其监控数据了。

因此，系统配置不当也是数据图产生中断的常见原因之一。当然，如果由于系统配置不当造成数据图中断，则我们通过监控项目列表页面或者数据库中 items 数据表里 errors 字段中所记录的内容是很容易发现的。

3. 被监控主机配置不当和性能瓶颈引起的数据缺失

我们知道，虽然 Zabbix 系统支持多达 13 种之多的监控数据采集方法，但是，这些监控数据采集方法中除了数据聚合和通过计算方法采集监控数据之外，其他的监控数据采集方法都需要在被监控主机（或其代理）上运行某些特定服务或脚本程序，才能使 Zabbix 系统采集到监控数据。这样，如果被监控主机（或其代理）上由于配置错误或者其他原因，导致这些特定服务或者脚本程序无法正常运行，那将必然会导致 Zabbix 服务器或者 Zabbix 服务器代理无法正常采集到监控数据。同样，如果被监控主机（或者其代理）遇到了硬件性能瓶颈，那么运行在被监控主机（或者其代理）上的特定服务或脚本程序，将无法及时接收和回应 Zabbix 服务器（或 Zabbix 服务器代理）的监控数据采集请求，这将同样会导致监控数据采集失败，从而导致监控数据的缺失。

4. Zabbix 服务器或 Zabbix 服务器代理性能瓶颈引起的数据缺失

前面介绍了，Zabbix 系统的管家进程可能会引起 Zabbix 系统中某些历史数据缺失。严格来说，我们的这个提法是不正确的。因为 Zabbix 系统的管家进程本身是不会引起历史数据缺失的，它是不会无缘无故地删除系统中不应该删除的历史数据的。而我们之所以说，Zabbix 系统的管家进程可能会引起 Zabbix 系统中历史数据的缺失，是因为在管家进程清理系统历史数据时，它会占用系统中大量的硬件资源，并因此会很容易造成 Zabbix 服务器（或 Zabbix 服务器代理）出现硬件性能瓶颈，从而使 Zabbix 系统暂时性地无法正常地采集到新的监控数据，或者无法正常地将所采集的新的监控数据写入到数据库中，最后导致 Zabbix 系统中某些历史数据缺失。因此，从本质上来说，Zabbix 系统管家进程所引起的 Zabbix 系统中某些历史数据的缺失，是由于管家进程导致 Zabbix 服务器（或其代理）暂时性地遇到硬件性能瓶颈所造成的。既然管家进程所引起的 Zabbix 服务器硬件性能瓶颈问题，可能会造成 Zabbix 系统丢失某些时段的历史数据，那么，当 Zabbix 服务器的硬件配置不高，而系统需要监控的对象又比较多时，纵使禁用 Zabbix 系统的管家进程，Zabbix 服务器同样也还有可能遇到硬件性能瓶颈问题，此时同样也会造成系统中某些时段的历史数据丢失。因此，Zabbix 服务器或 Zabbix 服务器代理的性能瓶颈也有可能引起 Zabbix 系统丢失历史数据。

5. 网络性能问题引起的数据缺失

我们知道，在 Zabbix 系统中，除了通过计算的方法和数据聚合的方法所采集的监控数据外，通过其他方法所采集的监控数据，都是先由特定服务或程序将它们从被监控主机或其代理上收集起来，然后通过一定的方法将它们通过网络传输到 Zabbix 服务器或 Zabbix 服务器代理上，因此，如果被监控主机（或其代理）与 Zabbix 服务器（或其代理）之间的网络性能很差，特别是当 Zabbix 服务器或其代理需要通过公网采集被监控主机上的监控数据时，就很容易造成因数据传输超时而导致所采集的监控数据丢失。因此，网络性能问题也是造成 Zabbix 系统数据丢失的一个很常见的原因。

9.6 本章小结

数据图中中文信息乱码的问题可能不仅仅是使用 Zabbix 系统的用户才会遇到的问题，其他监控系统的用户同样也可能会遇到这个问题。因此，本章一开始就介绍了如何解决这类问题。在介绍完如何解决中文信息乱码问题后，接着介绍了如何完善 Zabbix 系统的汉化效果。

通过本书的介绍，结合读者自己的学习和测试，相信绝大多数读者应该是认可 Zabbix 系统是一款功能强大，使用方便、灵活的开源监控系统的。但是，任何一款开源的监控系统都无法完全满足所有用户的所有需求。因此，在有能力的情况下，对 Zabbix 系统进行一些定制性的二次开发是可能的也是必要的。所以，本章以如何开发批量添加图表功能程序为例，详细介绍了如何对 Zabbix 系统进行二次开发。当然，这里所举的例子可能并不完全适用你，但是，通过这个例子，相信大多数读者对 Zabbix 系统二次开发的流程和相关知识还是有一定的认识和掌握的。开发了功能页面，我们当然希望将其整合到 Zabbix 系统的菜单中去。因此，在本章的接下来部分，详细介绍了如何在 Zabbix 系统中添加用户自定义的系统菜单项。

最后，详细探讨和分析了，我们在日常维护和管理 Zabbix 系统的过程中，可能会经常遇到的另一个问题——数据图断图的根本原因及其外部原因。

附录 A 触发器支持函数列表

以下是 Zabbix 系统中触发器表达式中可以使用的函数列表及其功能说明。实际上，这些函数在 Zabbix 系统中很多地方都可以使用。

函数	参数	支持的信息类型	说明
abschange	无	浮点型、整型、字符串型、文本型及日志型	返回监控项目上最新采集到的监控数据的绝对值，与前一次所采集到的监控数据的绝对值的比较情况。当两次监控数据绝对值相等时，返回0，否则返回1
avg	sec 或 #num	浮点型、整型	该函数用于统计系统针对某个监控项目在某个区间内所采集的监控数据的平均值。区间参数可以是以秒为单位（sec）的时间段，也可以是最近所采集到的数据个数（当表示系统最近所采集到的监控数据个数时，参数用#num形式。例如，#5表示系统最近所采集到的5个监控数据。该函数还支持第二个可选参数time_shift，它用于指定用于计算的数据集的时间起点，即以time_shift参数所指定的时刻为起点，求指定时间区间（或个数）内系统所采集的监控数据的平均值。例如，avg（3600,86400）表示求一天之前的一个小时内系统所采集的监控数据的平均值
change	无	浮点型、整型、字符串型、文本型及日志型	该函数返回系统最新采集到的监控数据与系统前一次采集到的监控数据的比较情况。当两次所采集的监控数据相等时，该函数返回值为0，否则，返回值为1
count	sec 或 #num	浮点型、整型、字符串型、文本型及日志型	该函数返回指定区间内，系统针对指定的监控项目所采集到的监控数据的个数。该函数的第一参数所表示的区间可以是以秒为单位的时间，也可以是表示数据个数的时间段（数值前面用#标识）。该函数可接受的第二个可选参数为匹配模式（pattern）参数。第三个可选参数是操作符参数（operator）。第四个可选参数是时间偏移参数（time_shift）。其中，该函数支持的操作符参数有：eq，表示等于；ne，表示不等于；gt，表示大于；ge，表示大于或等于；lt，表示小于；le，表示小于或等于；like，表示匹配指定的字符串模式。对于无符号的整型和浮点型数据，该函数支持的操作符有：eq（该函数默认操作符）、ne、gt、ge、lt、le；而对于信息类型是文本、字符串或日志类型的监控项目，该函数支持的操作符有：like（该函数默认操作符）、eq、ne等。以下是应用该函数的几个例子：

续表

函数	参数	支持的信息类型	说明
			<p>count(600), 返回最近10分钟内, 系统针对指定的监控项目所采集到的监控数据的个数</p> <p>count(600,12), 返回最近10分钟内, 系统针对指定的监控项目所采集到的监控数据中, 值等于12的监控数据的个数</p> <p>count(600,12,“gt”), 返回最近10分钟内, 系统针对指定的监控项目所采集到的, 数值大于12的监控数据的个数</p> <p>count(#10,12,“gt”), 返回系统针对指定的监控项目, 最近所采集到的10个监控数据中, 数值大于12的监控数据的个数</p> <p>count(600,12,“gt”,86400), 返回当前时间10分钟之前到24小时之内这段时间内, 系统针对指定的监控项目所采集到的监控数据中, 数值大于12的监控数据的个数</p> <p>count(600,,,86400), 返回当前时间10分钟之前到24小时之内这段时间内, 系统所采集到的监控数据的个数</p>
date	无	任何类型的监控项目	返回YYYYMMDD格式的Zabbix系统当前日期, 例如, 20140709
dayofmonth	无	任何类型的监控项目	返回月份的第几天。返回值的范围为1~31之间的正整数
dayofweek	无	任何类型的监控项目	返回星期几。返回值的范围为1~7, 其中1表示星期一, 7表示星期天
delta	sec 或 #num	浮点型、整型	返回指定时间段内, 系统针对指定的监控项目所采集到的监控数据中, 最大值和最小值之间的差值。时间段参数可以是以秒为单位(sec)的时间段, 也可以是以系统最近所采集到的监控数据个数所表示的时间段, 以数值前加上#号标识
diff	无	浮点型、整型、字符串型、文件型和日志型	返回系统针对指定监控项目所采集的, 最近一个监控数据与它的前一个监控数据的比较情况。如果两个值不同该函数返回1, 否则返回0
fuzzytime	sec	浮点型、整型	当系统针对指定的监控项目所采集到的监控数据中包含的时间戳与Zabbix系统当前的时间戳相差不大于N秒时, 则函数返回1; 否则返回0。该函数通常用于检查被监控主机上的系统时间与Zabbix服务器(或其代理)上系统的时间是否是同步的
iregexp	第一个参数是字符串, 第二参数是 sec 或 #num	字符串、日志或文本型	该函数的功能与regexp函数的功能是类似的, 只是该函数在执行字符串匹配时, 是忽略大小写的
last	sec 或 #num	浮点型、整型、字符串型和日志型	返回系统针对指定的监控项目最近所采集到的监控数据。当该函数的第一个参数是秒时, 则该参数被忽略。也就是说, 当以last(0)和last(10)两种形式调用该函数时, 其效果是一样的, 系统都是返回其针对指定的监控项目所采集的最后一个监控

续表

函数	参数	支持的信息类型	说明
			数据。而如果该函数的第一个参数是个数形式(用#加数字的形式表示),则其含义却不一样。例如, last(#3),表示返回系统针对指定的监控项目所采集的倒数第三个监控数据。该函数还支持以时间偏移参数作为其可选的第二个参数,例如 last(0,86400),表示返回24小时前的当前时刻,系统针对指定的监控项目所采集的最新监控数据
logevenid	string	日志型	返回系统针对指定的监控项目所采集的最新日志型数据中,事件ID是否匹配指定的正则表达式,如果匹配则返回1,否则返回0
logseverity	无	日志型	返回系统针对指定的监控项目所采集的最新日志型数据中,日志信息的级别。0表示默认级别,否则该函数返回值为N。其中,N为正整数。对于Windows事件日志,N的值与日志级别的对应关系为:1,信息;2,警告;4,错误;7,审计失败;8,审计成功。Zabbix系统从Windows事件日志信息中提取事件的级别信息
logsource	string	日志型	检查系统最新采集的日志型数据中日志来源是否与该函数参数所指定的字符串匹配。0表示不匹配,1表示匹配
max	sec 或 #num	浮点型、整型	返回指定时间段内,系统针对指定监控项目所采集的监控数据的最大值。时间段参数可以是以秒为单位(sec)所表示的时间段,也可以是以系统最近所采集到的监控数据个数所表示的时间段,以数值前加上#号来标识
min	sec 或 #num	浮点型、整型	返回指定时间段内,系统针对指定监控项目所采集到的监控数据的最小值。时间段参数可以是以秒为单位(sec)所表示的时间段,也可以是以系统最近所采集到的数据个数所表示的时间段,以数值前加上#号来标识
nodata	sec	任何类型的数据	如果系统在指定时间段内,针对指定的监控项目未采集到任何监控数据,则返回1,否则返回0。时间段参数必须是不小于30的正整数
now	无	任何类型的数据	返回UNIX时间戳,即从1970年1月1日0时0分0秒到当前时刻的总秒数
prev	无	浮点型、整型、字符串型和日志型	返回系统针对指定的监控项目所采集到的,最近一个数据的前一个数据,即该函数的返回值与 last(#2)函数的返回值是相同的
regexp	第一个参数是字符串,第二个参数是sec或#num	字符串、日志或文本型	该函数用于检查系统针对指定的监控项目,最新采集的监控数据是否匹配指定的正则表达式。其中,该函数的第一个参数就是要匹配的正则表达式。而第二个参数是一个可选参数,它可以是表示时长的秒数,也可以是所采集到的监控数据的个数,以数值前面加#号来标识。当在该函数的第二个参数所指定的时间(或个数)范围内,系统针对指定的监控项目采集到了多个监控数据,则

续表

函数	参数	支持的信息类型	说明
			该函数会逐个对这些监控数据进行匹配。当发现有匹配成功的数据时，该函数返回1，否则返回0。该函数与iregexp函数最大的不同之处在于，该函数在对数据进行匹配时是区分大小写的，即对大小写是敏感的
str	第一个参数是字符串，第二个参数是 sec 或 #num	字符串、日志或文本型	该函数用于从系统针对指定的监控项目最新所采集的监控数据中，查找指定的字符串。该函数第一个参数指定需要查找的字符串。而第二个参数是一个可选参数，它可以是表示时长的秒数，也可以是表示监控数据的个数，以数值前面加#号来标识。当在该函数的第二个参数所指定的时间（或个数）范围内，系统针对指定的监控项目采集到了多个监控数据，则该函数会逐个针对这些监控数据进行查找。当在指定的数据中查找到了指定的字符串时，则该函数返回1，否则返回0
strlen	sec 或 #num	字符串、日志或文本型	该函数返回系统针对指定的监控项目，最新所采集到的监控数据的字符长度。该函数参数的含义与last函数参数的含义是一致的。比如，strlen(0)与strlen(#1)功能是一样的，都表示返回系统针对指定的监控项目所采集的最后一个监控数据的字符长度；而strlen(#3)则表示，系统针对指定的监控项目所采集的倒数第三个监控数据的字符长度；类似的，strlen(0,86400)则表示返回系统针对指定监控项目在24小时前的当前时刻，所采集到的最新监控数据的字符长度
sum	sec 或 #num	浮点型或整型	该函数用于计算，在指定时间段内，系统针对指定监控项目所采集到的监控数据之和。该函数的参数可以是表示时长的秒数，也可以是表示监控数据的个数，以数值前面加#号来标识
time	无	任何类型的数据	该函数返回HHMMSS格式的当前时间，例如114705

- 提示：1. 所有的函数返回值都是数值型的。
2. 当函数参数是字符串时，该字符串需要用双引号引起来，否则系统会报解析错误。

附录 B Zabbix 系统中的单位符号

在 Zabbix 系统中有一些我们经常会使用的单位符号，这里对其进行了整理并列在下面，以方便读者查阅。

时间后缀符号

- s——秒
- m——分
- h——时
- d——日
- w——星期

时间后缀符号可以在以下这些地方得到支持：

- 触发器表达式中（常量和函数参数中）。
- Zabbix 内部监控项目 `zabbix[queue,<from>,<to>]` 的参数中。
- 聚合数据采集方法中的最后一个参数。

前缀符号

- K——千（Kilo）
- M——兆（Mega）
- G——吉（Giga）
- T——太（Tera）

需要注意的是，在 Web 组件的前台页面上显示字节（B）、字节每秒（B/s）时，上述这些单位的进阶是 1000，即 $1K=1000$ 。而当显示其他单位类型的数据时，上述些单位进阶是 2 的 10 次方，即 $1K=1024$ 。

当系统在 Web 组件的前台页面上要显示的数据大于 1024 太（T）时，系统可能还会使用以下这些更高阶的前缀符号：

- P——拍（peta）
- E——艾（exa）
- Z——泽（zeta）
- Y——尧（yotta）

合理地使用单位符号或前缀符号，可以使我们书写出的触发器表达式更易于理解和方便日常维护，例如下面这三个触发器表达式的例子：

```
{host:zabbix[proxy,zabbix_proxy,lastaccess]}>120
{host:system.uptime[].last(0)}<86400
{host:system.cpu.load.avg(600)}<10.
```


如果将其修改成下面这种形式，不但可以更容易理解，而且还能减少因为少写一位数字而导致的错误：

```
{host:zabbix[proxy,zabbix_proxy,lastaccess]}>2m
{host:system.uptime.last(0)}<1d
{host:system.cpu.load.avg(10m)}<10.
```

附录 C Zabbix Agent 监控项目关键字列表

我们知道，通过被监控设备代理组件采集监控数据的监控项目，其合法和可用的关键字是在开发 Zabbix 系统时写死的。因此，我们在配置这种类型的监控项目时，不可以随意填写其关键字，其关键字必须是下表所列的关键字之一。

关键字	返回值	参数	描述
agent.hostname	返回被监控主机上被监控设备代理配置文件中所配置的主机名	无	
agent.ping	用于检查指定的被监控设备代理是否可用。如果被监控设备代理可用，则返回1，否则不返回任何数据	无	该关键字所对应的监控项目可以与nodata()函数联合，用于检查被监控设备代理是否可用
agent.version	返回被监控主机上所运行的被监控设备代理组件的版本号	无	
kernel.maxfiles	返回操作系统中所设置的可以打开的最大文件句柄数		
kernel.maxproc	返回操作系统可以支持的最大进程数		
log[file,<regexp>,<encoding>,<maxlines>,<mode>]	返回被监控日志文件中，成功匹配正则表达式的所有行的日志信息	file——被监控的日志文件的完整路径，例如/tmp/zabbix_server.log等。对于该文件，被监控设备代理至少需要有读取它的权限。 regexp——用于匹配的正则表达式。 encoding——用于指定日志文件的编码。 maxlines——被监控设备代理每ns发送给Zabbix服务器端或Zabbix服务器代理端最大日志信息行数。该参数的值会覆盖掉zabbix_agentd.conf文件中MaxLinesPerSecond配置项的值。 mode——模式，可能的值为all（默认值），skip（跳过老的数据）	需要注意的是，该关键字所对应的监控项目，需要配置成被监控设备代理主动采集监控数据方式。如果该关键字中所指定的日志文件不存在，或者被监控设备代理组件进程没权限读取它，则对应的监控项目将会变成不被支持状态 例子：log["/var/log/nginx/zabbix.log" ,"192.168.5.139", utf8,10]

续表

关键字	返回值	参数	描述
<code>logrt[file_pattern, <regexp>,<encoding> ,<maxlines>,<mode>]</code>	返回被监控日志文件中,成功匹配正则表达式的所有行的日志信息。该关键字与上面 <code>log[……]</code> 关键字的含义和作用类似,只是该关键字所对应的监控项目可以支持对轮转日志文件进行监控	<code>file_pattern</code> ——被监控日志文件的绝对路径和它的文件名正则表达式。 <code>regexp</code> ——所需要匹配的正则表达式。 <code>encoding</code> ——用于指定日志文件的编码。 <code>maxlines</code> ——用于指定被监控设备代理每ns发送给Zabbix服务器端或其代理端最大日志信息行数。该参数的值会覆盖掉zabbix_agentd.conf文件中MaxLinesPerSecond配置项的值。 <code>mode</code> ——模式,可选的值有all(默认值)、skip(跳过老的数据)	需要注意的是,该关键字所对应的监控项目,需要被配置成被监控设备代理主动采集数据方式 例子: <code>logrt["/var/log/nginx/^logfile[0-9]{1,3}\$", "192.168.5.139",utf8,10]</code> ,可以监控logfile0、logfile001、logfile10等文件名的日志文件
<code>net.dns[<ip>,<zone>,<type>,<timeout>,<count>]</code>	返回DNS服务是否为可用状态。如果DNS服务不可用,或者查询时DNS服务器没有响应则返回0,否则返回1	<code>ip</code> ——DNS服务器的IP地址。该参数如果留空,则系统会检测被监控主机上所配置的默认DNS服务器上DNS服务的状态。 <code>zone</code> ——用于指定被检测DNS的域。 <code>type</code> ——用于指定被检测的记录类型。 <code>timeout</code> (Windows平台下忽略该参数)——用于指定检测的超时时间,默认值是1秒。 <code>count</code> (Windows平台下忽略该参数)——用于指定检测的尝试次数,默认是2次	例子: <code>net.dns[8.8.8.8,zabbix.com,MX,2,1]</code> 。 可能的记录类型,即type参数值有: ANY、A、NS、CNAME、MB、MG、MR、PTR、MD、MF、MX、SOA、NULL、WKS、HINFO、MINFO、TXT、SRV等
<code>net.dns.record[<ip>,<zone>,<type>,<timeout>,<count>]</code>	返回被执行的DNS查询的结果。如果执行成功,则返回带查询类型的字符串信息	<code>ip</code> ——用于指定DNS服务器IP地址。如果该参数留空,则系统查询被监控主机上所配置的默认DNS服务器 <code>zone</code> ——用于指定查询的DNS域 <code>type</code> ——用于指定查询的记录类型 <code>timeout</code> (Windows平台下忽略该参数)——用于指定查询的超时时间,默认值	例子: <code>net.dns.record[8.8.8.8,zabbix.com,MX,2,1]</code> ,返回的数据信息为:“zabbix.com MX 0 mail.zabbix.com”。可能的记录类型,即type参数值有: ANY、A、NS、CNAME、MB、MG、MR、PTR、MD、MF、MX、SOA、NULL、WKS、HINFO、MINFO、TXT、SRV等

续表

关键字	返回值	参数	描述
		是1秒 count（Windows平台下忽略该参数）——用于指定查询的尝试次数，默认是2次	
net.if.collisions[if]	返回指定网卡上，所检测到的帧发送时通信线路上所发现的碰撞次数	if——网卡名称，例如eth0、em1等。	帧碰撞是带冲突检测的载波侦听多路访问技术中的一个概念，读者如果有兴趣可以参考相关计算机网络管理方面的书籍
net.if.discovery	返回JSON格式的被监控主机上所配置的网卡信息。该关键字用于低级自动发现功能		
net.if.in[if,<mode>]	返回指定网卡上所接收到的流量统计，返回值为整数	if——网卡名称，例如eth0、eth1等。 mode——可选的值有：字节数（bytes），返回值代表的是指定网卡上所接收到的数据总位数（mode参数的默认值是bytes）；包数（packets），返回指定网卡上，所接收到的总包数；错误数（errors），返回指定网卡上接收到的失败的数据数；丢弃数（dropped），返回指定网卡上，被丢弃的数据包数。	
net.if.out[if,<mode>]	返回指定网卡上所发送的流量统计，返回值为整数	if——网卡名称，例如eth0、eth1等 mode——可选的值有：字节（bytes），返回值代表指定网卡上所发送的数据总位数(mode参数的默认值是bytes)；包数（packets），返回指定网卡上所发送的总包数；错误数（errors），返回指定网卡上发送失败的数据数；丢弃数（dropped），返回指定网卡上被丢弃的数据包数	
net.if.total[if,<mode>]	返回指定网卡上接收和发送总流量统计，返回值为整数	if——网卡名称，例如eth0、eth1等 mode——可选的值有：字节（bytes），返回值代表指定网卡上所发送的数据总	

续表

关键字	返回值	参数	描述
		位数 (mode参数的默认值是bytes); 包数 (packets), 返回指定网卡上所发送的总包数; 错误数 (errors), 返回指定网卡上发送失败的数据数; 丢弃数 (dropped), 返回指定网卡上被丢弃的数据包数	
net.tcp.listen[port]	返回指定的TCP端口是否处于侦听状态。如果指定的TCP端口未处于侦听状态则返回0, 否则返回1	port——指定需要检测的TCP端口号	该关键字所对应的监控项目常被用于检测某个服务是否可用, 但是需要注意的是, 因为检测是通过被监控设备代理执行的, 所以检测实际上相当于在被监控主机上执行
net.tcp.port[<ip>,port]	返回指定的TCP端口是否处于可以被连接状态。如果不能被连接则返回0, 否则返回1	ip——被连接的IP地址, 默认值是127.0.0.1。这个参数一般被指定为所检测的端口侦听的IP地址。 port——被测试的TCP端口号	该关键字所对应的监控项目常被用于检测某个服务是否可用, 但是需要注意的是, 因为检测是通过被监控设备代理执行的, 所以检测实际上相当于在被监控主机上执行
net.tcp.service[service,<ip>,<port>]	返回指定的服务是否正在运行, 并且是否处于可以接受连接的状态。当服务不可用时, 返回0; 否则返回1	service——服务名称, 可以是ssh、ntp、ldap、smtp、ftp、http、pop、nntp、imap、tcp、https和telnet中的任何一个。 ip——被检测的IP地址, 默认值是127.0.0.1。 port——服务的端口号, 如果不填则表示对应服务的默认端口号	例子: net.tcp.service[ftp,,45] 用于检查ftp服务是否可用, 端口号为45
net.tcp.service.perf[service,<ip>,<port>]	返回指定服务的性能数据。当返回值为0时, 则表示对应服务不可用, 否则返回系统连接到指定服务所花的秒数	service——服务名称, 可以是ssh、ntp、ldap、smtp、ftp、http、pop、nntp、imap、tcp、https和telnet中的任何一个 ip——被检测的IP地址, 默认值是127.0.0.1 port——服务的端口号, 如果不填则表示对应服务的默认端口号	
net.udp.listen[port]	返回指定的UDP端口是否处于侦听状态, 如果未侦听则返回0, 否则返回1	port——指定需要检测的端口号	

续表

关键字	返回值	参数	描述
proc.mem[<name>, <user>, <mode>, <cmdline>]	返回指定用户运行的进程所占用的内存数，单位是字节	name——进程名，默认值是“all processes” user——用户名，默认值是“all users” mode——可选的值有avg、max、min、sum（默认值）等 cmdline——使用命令行内容过滤	
proc.num[<name>, <user>, <state>, <cmdline>]	返回指定用户运行且处于指定状态下的进程数	name——进程名，默认值是“all processes” user——用户名，默认值是“all users” state——状态，可选值有all（默认值）、run、sleep、zomb等 cmdline——使用命令行内容过滤	
sensor[device,sensor, <mode>]	读取指定的硬件传感器数据	device——设备名称，如果使用了mode参数，则设备名称可以是一个正则表达式 sensor——传感器名称，如果使用了mode参数，则传感器名称可以是一个正则表达式； mode——可选的值有：avg、max和min等。如果省略mode参数，则设备名称和传感器名称参数都被系统视为单个名称，而不是将其当作正则表达式看待	
system.boottime	返回系统被启动时的时间戳		
system.cpu.intr	返回硬件中断数		
system.cpu.load[<cpu>, <mode>]	返回系统中CPU的负载数，它是一个浮点数	cpu——可选的值有：all（默认值）、percpu（总负载数除以可用CPU核数所得到的浮点数） mode——可选的值有：avg1（一分钟平均负载，默认值）、avg5（5分钟平均负载）、avg15（15分钟平均负载）	
system.cpu.num[<type>]	返回系统中可用的CPU数	type——可选的值有online（默认值）、max	

续表

关键字	返回值	参数	描述
system.cpu.switches	返回系统中上下文交换次数		
system.cpu.util[<cpu>,<type>, <mode>]	返回CPU的使用率	cpu——CPU数（默认值是所有CPU）； type——可选的值有：idle、nice、user（默认值）、system（windows平台下）、iowait、interrupt、softirq、steal等； mode——可选的值有avg1（一分钟平均使用率，默认值）、avg5（5分钟平均使用率）、avg15（15分钟平均使用率）	
system.hostname[<type>]	返回系统的主机名	type——该参数只有在Windows平台下可用，其他平台下忽略	
system.hw.chassis[<info>]	返回系统主板信息	Info——可选值有full（默认值）、model、serial、type或vendor	需要注意：如果在被监控主机上配置了该关键字对应的监控项目，则对应被监控主机上的被监控设备代理组件需要以root用户运行
system.hw.cpu[<cpu>, <info>]	返回系统中CPU信息	cpu——CPU号或者all（默认值） info——可选值有：full（默认值）、curfreq、maxfreq、model或vendor	
system.hw.devices[<type>]	返回PCI或USB设备列表信息	type——pci（默认值）或usb	
system.hw.macaddr[<interface>, <format>]	返回系统中网卡的MAC地址列表	interface——可选值有all（默认值）或者正则表达式 format——可选值有full（默认值）或short	
system.localtime[<type>]	返回系统时间	type——可选值有utc（默认值），返回utc格式的时间；local,返回yyyy-mm-dd, hh:mm:ss.nn格式的本地时间	
system.run[command, <mode>]	在被监控主机上执行指定的命令，并返回该命令序列被成功执行后的返回信息	command——用于指定将要被执行的命令 mode——可选值有：wait（等待命令执行完成）、nowait（不等待命令执行完成）	被执行的命令所返回的信息必须是小于64KB的文本信息

续表

关键字	返回值	参数	描述
system.stat[resource, <type>]	返回虚拟内存的统计信息	参数略	该关键字只在AIX系统下使用
System.sw.arch	返回系统的体系架构信息	无	例子： system.sw.arch 返回 i686
system.sw.os[<info>]	返回操作系统信息	full——可选值有full（默认值）、short和name	
system.sw.packages[<package>,<manager>,<format>]	返回系统中安装的软件包列表	package——可选值有all（默认值）或者正则表达式； manage——可选值有all（默认值）或者包管理器名称； format——可选值有full（默认值）和short	
system.swap.in[<device>,<type>]	返回系统中交换分区换入（即信息由设备读入到内存）信息统计	device——用作交换的设备（默认值取all） type——可选的值有count、sectors和pages	
System.swap.out[<device>,<type>]	返回系统中交换分区换出（即信息由内存写入到交换设备上）信息统计	Device——用作交换的设备（默认值取all） type——可选的值有count、sectors和pages	
system.swap.size[<device>,<type>]	返回交换空间使用情况统计信息	device——用作交换的设备（默认值取all） type——可选的值有free（返回空闲交换空间，默认值）、pfree（返回空闲交换空间，百分比格式）、pused（返回已使用交换空间，百分比格式）、total（总的交换空间）和used（已使用的交换空间）	例子： system.swap.size[,pfree] 返回百分比格式的空闲交换空间
system.uname	返回详细的主机信息	无	
system.uptime	返回系统在线时间，单位是秒		
system.users.num	返回登录到系统中的用户数		
vfs.dev.read[<device>,<type>,<mode>]	返回磁盘的读数据统计信息。如果参数type取sectors、operations、bytes中的一个，则返回值是整数；如果type取sps、ops和bps中的一个，则返回值是浮点数	device——磁盘设备名（默认值是all）； type——可选的值有secotrs、operations、bytes、sps和ops mode——可选的值有avg1（一分钟内的平均数）、avg5（5分钟内的平均数）、avg15（15分钟内的平均数）	注意，只有当type参数取sps、ops和bps中的一个时，才需要指定mode参数值

续表

关键字	返回值	参数	描述
<code>vfs.dev.write[<device>,<type>, <mode>]</code>	返回磁盘的写数据统计信息。如果参数type取sectors、operations、bytes中的一个，则返回值为整数；如果type取sps、ops和bps中的一个，则返回值为浮点数	device——磁盘设备名（默认值是all）； type——可选的值有secotrs、operations、bytes、sps和ops。 mode——可选的值有avg1（一分钟内的平均数）、avg5（5分钟内的平均数）、avg15（15分钟内的平均数）	注意，只有当type参数取sps、ops和bps中的一个时，才需要指定mode参数值
<code>vfs.file.cksum[file]</code>	计算并返回指定文件的检验码	file——所要计算的文件的完整路径	
<code>vfs.file.contents[file,<encoding>]</code>	读取并返回指定文件的内容，如果该文件为空，或者仅包括回车和换行符，则该关键字返回EOF，否则返回所读取的文件内容	file——所需要读的文件完整路径	
<code>vfs.file.exists[file]</code>	判断指定文件是否存在。如果指定的文件存在，且为普通文件或连接文件（包括软连接和硬连接文件），则返回1，否则返回0	file——所需要判断的文件完整路径。	
<code>vfs.file.md5sum [file]</code>	计算并返回指定文件的MD5哈希数	file——所需要计算的文件的完整路径	
<code>vfs.file.regexp[file, regexp,<encoding>]</code>	在指定文件中查找指定的字符串。返回包含指定字符串内容的整行内容，或者如果在指定文件中没有查到指定的字符串内容，则返回EOF	file——完整的文件路径 regexp——正则表达式 encoding——编码	该关键字所对应的监控项目，仅返回首次发现指定字符串内容所在行的信息
<code>vfs.file.regmatch[file,regexp, <encoding>]</code>	在指定文件中查找指定的字符串。如果找到则返回1，否则返回0	file——完整的文件路径 regexp——正则表达式 encoding——编码	
<code>vfs.file.size[file]</code>	返回指定文件的大小，单位是字节	file——完整的文件路径	运行被监控设备代理组件的用户需要对指定文件有读取权限
<code>vfs.file.time[file, <mode>]</code>	获取并返回指定文件的时间信息，返回值是Unix格式的时间戳	file——完整的文件路径； mode——可选的值有modify（默认值，文件的修改时间）、access（文件的最后一次访问时间）和change（文件最后一次修改时间）	

续表

关键字	返回值	参数	描述
vfs.fs.discovery	发现并返回系统中，已经挂载的文件系统列表信息。该关键字一般使用在低级自动发现功能中	无	
vfs.fs.inode[fs,<mode>]	返回指定文件系统的inode数	fs——文件系统 mode——可选的值有total（默认值）、free、used、pfree（空闲的inode数，百分比形式）、pused（已使用的inode数，百分比形式）	
Vfs.fs.size[fs,<mode>]	返回磁盘空间使用情况	fs——文件系统； mode——可选的值有total（默认值）、free、used、pfree（空闲空间，百分比形式）、pused（已使用空间，百分比形式）	
Vm.memory.size[<mode>]	返回系统中内存使用情况	mode——可选值有total（默认值）、active、anon、buffers、cached、exec、file、free、inactive、pinned、shared、wired、used、pused、available、pavailable	
web.page.get[host,<path>,<port>]	返回Web页面的源代码信息	host——主机名 path——HTML文档的路径 port——端口号（默认值是80）	如果指定Web页面的源代码抓取失败，则返回EOF
web.page.perf[host,<path>,<port>]	返回Web页面的加载时间信息，返回值是以秒为单位的时间	host——主机名 path——HTML文档的路径 port——端口号（默认值是80）	如果获取时间信息失败，则返回数值0
web.page.regex[host,<path>,<port>,<regex>,<length>]	返回指定Web页面上，首次匹配成功指定正则表达式的内容	host——主机名 path——HTML文档的路径 port——端口号（默认值是80） regex——正则表达式 length——指定返回内容的最大字符长度	如果在指定的Web页面上，没有发现有匹配成功的字符串，则返回EOF

附录 D Zabbix 支持的宏变量列表

在 Zabbix 系统中，可以支持的宏变量如下表所示。其中，1、2、3……标识的列表示的是宏变量出现的位置，它们所代表的位置为：

- 1——通知和命令行中
- 2——自动发现的通知中
- 3——自动注册通知中
- 4——全局脚本中
- 5——监控项目关键字的参数中
- 6——拓扑图的标签中
- 7——拓扑图的 URL 地址中
- 8——触发器表达式中
- 9——触发器名称中
- 10——监控项目名称中
- 11——接口的 IP 地址或 DNS 主机名中
- 12——数据库监控中附加参数或 SSH 和 Telnet 脚本中

而表中“X”符号表示，宏变量在对应的位置是被支持的；否则表示对应宏变量在对应的位置不被支持。而形如{MACRO<1-9>}格式的宏变量中的数字，用于指代在触发器表达式中出现的第几个主机。例如，{HOST.IP1}、{HOST.IP2}和{HOST.IP3}分别代表在触发器表达式中出现的第一个、第二个和第三个主机所对应的 IP 地址。因此，在系统对触发器表达式进行计算时，相应的宏变量将会被对应主机的 IP 地址所替换。

宏变量	1	2	3	4	5	6	7	8	9	10	11	12	描述
{DATE}	X	X	X										代表yyyy.mm.dd格式的当前时间
{DISCOVERY.DEV ICE.IPADDRESS}		X											代表被自动发现设备的IP地址，无论对应的设备是否已被添加到系统中，该宏变量都可用
{DISCOVERY.DEV ICE.DNS}		X											代表被自动发现设备的DNS主机名，无论对应的设备是否已被添加到系统中，该宏变量都可用
{DISCOVERY.DEV ICE.STATUS}		X											代表被自动发现主机的状态，可能值为UP或DOWN
{DISCOVERY.DEV ICE.UPTIME}		X											代表某台设备自上次自动发现状态改变以来，所经历的时间长度，例如1小时29分等。需要注意的是，当某台

续表

宏变量	1	2	3	4	5	6	7	8	9	10	11	12	描述
													设备自动发现状态为DOWN状态时，则该宏变量代表对应设备自从UP状态转变为DOWN状态时，到当前时间所经历的时长
{DISCOVERY.RULE.NAME}		X											代表发现某台设备或服务时，系统所使用的自动发现规则的名称
{DISCOVERY.SERVICE.NAME}		X											被发现的服务的名称，例如HTTP等
{DISCOVERY.SERVICE.PORT}		X											被发现的服务的端口，例如80等
{DISCOVERY.SERVICE.STATUS}		X											被发现服务的状态，可能值为UP或DOWN
{DISCOVERY.SERVICE.UPTIME}		X											代表某个被自动发现的服务，自上次自动发现状态改变以来所经历的时长。例如，1h29m等。如果服务处于DOWN状态，则该宏变量所代表的时长代表对应服务持续处于DOWN状态的时长
{ESC.HISTORY}	X												代表步骤升级历史。如该步骤之前所发送的信息日志的状态等
{EVENT.ACK.HISTORY}	X												代表“问题”事件的确认日志
{EVENT.ACK.STATUS}	X												代表事件的确认状态，可能值为yes或no
{EVENT.AGE}	X	X	X										代表事件存续时长
{EVENT.DATE}	X	X	X										代表事件所发生的日期
{EVENT.ID}	X	X	X										代表事件ID
{EVENT.TIME}	X	X	X										代表事件所发生的时间
{HOST.CONN<1-9>}	X			X	X ¹	X			X		X	X ⁴	代表主机的IP或主机的DNS主机名，依赖于主机配置 ²
{HOST.DNS<1-9>}	X			X	X ¹	X			X		X	X ⁴	代表主机的DNS主机名 ²
{HOST.HOST<1-9>}	X		X	X	X ¹	X			X		X	X ⁴	代表主机名
{HOST.ID}							X						代表主机ID
{HOST.IP<1-9>}	X		X	X	X ¹	X			X		X	X ⁴	代表主机的IP地址
{HOST.NAME<1-9>}	X		X	X	X ¹	X			X		X	X ⁴	代表主机的显示名称
{HOST.PORT}			X										代表在被监控主机上运行的被监控设备代理所侦听的端口号
{HOSTGROUP.ID}							X						代表主机组ID

续表

宏变量	1	2	3	4	5	6	7	8	9	10	11	12	描述
{INVENTORY.ALIAS<1-9>}	X												代表主机资产中匿名 (Alias) 字段中的内容
{INVENTORY.ASSET.TAG<1-9>}	X												代表主机资产中资产标签 (Asset Tag) 字段中的内容
{INVENTORY.CHASSIS<1-9>}	X												代表主机资产中硬件信息 (Chassis) 字段中的内容
{INVENTORY.CONTACT<1-9>}	X												代表主机资产中联系人 (Contact) 字段中的内容
{INVENTORY.CONTRACT.NUMBER<1-9>}	X												代表主机资产中合同编号 (Contract Number) 字段中的内容
{INVENTORY.DEPLOYMENT.STATUS<1-9>}	X												代表主机资产中部署状态 (Deployment) 字段中的内容
{INVENTORY.HARDWARE<1-9>}	X												代表主机资产中硬件 (Hardware) 字段中的内容
{INVENTORY.HARDWARE.FULL<1-9>}	X												代表主机资产中详细硬件 (Hardware Full detail) 字段中的内容
{INVENTORY.HOST.NETMASK<1-9>}	X												代表主机资产中主机子网掩码 (Host subnet mask) 字段中的内容
{INVENTORY.HOST.NETWORKS<1-9>}	X												代表主机资产中主机网络 (Host networks) 字段中的内容
{INVENTORY.HOST.ROUTER<1-9>}	X												代表主机资产中主机路由 (Host router) 字段中的内容
{INVENTORY.HW.ARCH<1-9>}	X												代表主机资产中硬件架构 (HW architecture) 字段中的内容
{INVENTORY.HW.DATE.DECOMM<1-9>}	X												代表主机资产中硬件报废日期 (Date HW decommissioned) 字段中的内容
{INVENTORY.HW.DATE.EXPIRY<1-9>}	X												代表主机资产中硬件保修截止日期 (Date hardware maintenance expires) 字段中的内容
{INVENTORY.HW.DATE.INSTALL<1-9>}	X												代表主机资产中硬件安装日期 (Date hardware installed) 字段中的内容
{INVENTORY.HW.DATE.PURCHASE<1-9>}	X												代表主机资产中硬件购买日期 (Date hardware purchased) 字段中的内容

续表

宏变量	1	2	3	4	5	6	7	8	9	10	11	12	描述
{INVENTORY.INSTALLER.NAME<1-9>}	X												代表主机资产中安装器名称 (Installer name) 字段中的内容
{INVENTORY.LOCATION<1-9>}	X												代表主机资产中位置 (Location) 字段中的内容
{INVENTORY.LOCATION.LAT<1-9>}	X												代表主机资产中位置纬度 (Location latitude) 字段中的内容
{INVENTORY.LOCATION.LON<1-9>}	X												代表主机资产中位置经度 (Location longitude) 字段中的内容
{INVENTORY.MACADDRESS.A<1-9>}	X												代表主机资产中网卡物理地址A (MAC address A) 字段中的内容
{INVENTORY.MACADDRESS.B<1-9>}	X												代表主机资产中网卡物理地址B (MAC address B) 字段中的内容
{INVENTORY.MODEL<1-9>}	X												代表主机资产中模型 (Model) 字段中的内容
{INVENTORY.NAME<1-9>}	X												代表主机资产中名称 (Name) 字段中的内容
{INVENTORY.NOTES<1-9>}	X												代表主机资产中备注 (Notes) 字段中的内容
{INVENTORY.OOB.IP<1-9>}	X												代表主机资产中OOB IP地址字段中的内容
{INVENTORY.OOB.NETMASK<1-9>}	X												代表主机资产中OOB子网掩码字段中的内容
{INVENTORY.OOB.ROUTER<1-9>}	X												代表主机资产中OOB路由字段中的内容。
{INVENTORY.OS<1-9>}	X												代表主机资产中操作系统字段中的内容
{INVENTORY.OS.FULL<1-9>}	X												代表主机资产中详细操作系统信息 (OS Full details) 字段中的内容
{INVENTORY.OS.SHORT<1-9>}	X												代表主机资产中操作系统简略信息 (OS Short) 字段中的内容
{INVENTORY.SERIALNO.A<1-9>}	X												代表主机资产中序列号A (Serial number A) 字段中的内容
{INVENTORY.SERIALNO.B<1-9>}	X												代表主机资产中序列号B (Serial number B) 字段中的内容

续表

宏变量	1	2	3	4	5	6	7	8	9	10	11	12	描述
{INVENTORY.SITE .ADDRESS.A<1-9>}	X												代表主机资产中位置信息A (Site address A) 字段中的 内容
{INVENTORY.SITE .ADDRESS.B<1-9>}	X												代表主机资产中位置信息B (Site address B) 字段中的 内容
{INVENTORY.SITE .ADDRESS.C<1-9>}	X												代表主机资产中位置信息C (Site address C) 字段中的 内容
{INVENTORY.SITE .CITY<1-9>}	X												代表主机资产中城市 (Site city) 字段中的内容
{INVENTORY.SITE .COUNTRY<1-9>}	X												代表主机资产中国家 (Site country) 字段中的内容
{INVENTORY.TAG <1-9>}	X												代表主机资产中标签 (Tag) 字段中的内容
{INVENTORY.TYP E<1-9>}	X												代表主机资产中类型 (Type) 字段中的内容
{INVENTORY.TYP E.FULL<1-9>}	X												代表主机资产中类型详情 (Type Full details) 字段中 的内容
{INVENTORY.VEN DOR<1-9>}	X												代表主机资产中供应商 (Type Full details) 字段中 的内容
{ITEM.ID<1-9>}	X												代表引起发送报警信息的触 发器中第N个监控项目的数 字ID号
{ITEM.DESRIPTI ON<1-9>}	X												代表引起发送报警信息的触 发器中, 第N个监控项目的描 述
{ITEM.KEY<1-9>}	X												代表引起发送报警信息的触 发器中, 第N个监控项目的关 键字
{ITEM.KEY.ORIG< 1-9>}	X												代表引起发送报警信息的触 发器中, 第N个监控项目的原 始关键字 (即关键字中所包 含的宏变量不被替换)
{ITEM.LASTVALU E<1-9>}	X								X				代表引起发送报警信息的触 发器中, 第N个监控项目所采 集到的最新的监控数据
{ITEM.NAME<1-9>}	X												代表引起发送报警信息的触 发器中第N个监控项目的名称

续表

宏变量	1	2	3	4	5	6	7	8	9	10	11	12	描述
{ITEM.NAME.ORG<1-9>}	X												代表引起发送报警信息的触发器中，第N个监控项目的原始名称（即监控项目名称中的宏变量不被替换）
{ITEM.VALUE<1-9>}	X								X				1. 当该宏变量使用在触发器中时，则它代表引起发送报警信息的触发器中，第N个监控项目所采集的最新监控数据。此时，它与{ITEM.LASTVALUE}宏变量的作用是相同的 2. 如果该宏变量应用在通知和命令行中，则该宏变量的值为对应事件所对应的历史数据。如果对应历史数据被删除，或者未保留历史数据，则该宏变量的值为UNKNOWN
{MAP.ID}							X						代表网络拓扑图的ID号
{NODE.ID<1-9>}	X	X	X										代表分布式节点的ID号
{NODE.NAME<1-9>}	X	X	X										代表分布式节点的名称
{PROXY.NAME<1-9>}	X	X	X										代表引起发送报警信息的触发器中，第N个监控项目所对应的Zabbix服务器代理名称
{TIME}	X	X	X										代表hh:mm:ss格式的当前时间
{TRIGGER.DESCRPTION}	X												代表触发器描述
{TRIGGER.EVENTS.ACK}	X					X							如果该宏变量使用在拓扑图的标签中，则其代表与拓扑图元素相关的已确认事件的数量；而如果该宏变量被使用在通知或命令行中，则该宏变量代表产生事件所对应的触发器
{TRIGGER.EVENTS.PROBLEM.ACK}	X					X							代表触发器所对应的已被确认的问题（PROBLEM）事件的数量
{TRIGGER.EVENTS.PROBLEM.UNACK}	X					X							代表所有触发器所对应的未被确认的问题（PROBLEM）事件的数量
{TRIGGER.EVENTS.UNACK}	X					X							如果这个宏变量使用在拓扑图的标签中，则其代表与拓扑图元素相关的未被确认的事件的数量；而如果该宏变

续表

宏变量	1	2	3	4	5	6	7	8	9	10	11	12	描述
													量被使用在通知或命令行中，则该宏变量代表产生事件所对应的触发器
TRIGGER.PROBLEM.EVENTS.PROBLEM.ACK}						X							该宏变量代表仍处于问题（PROBLEM）状态下的触发器所对应的事件中，已被确认的事件数量
{TRIGGER.PROBLEM.EVENTS.PROBLEM.UNACK}						X							该宏变量代表仍处于问题（PROBLEM）状态下的触发器所对应的事件中，未被确认的事件数量
{TRIGGER.EXPRESSION}	X												代表触发器表达式
{TRIGGER.ID}	X						X						代表触发器ID
{TRIGGER.NAME}	X												代表触发器名称
{TRIGGER.NAME.ORIG}	X												代表触发器的原名称（即未替换名称中宏变量的触发器名称）
{TRIGGER.NSEVERITY}	X												代表触发器级别的数字形式，其值的范围为0~5的整数
{TRIGGER.SEVERITY}	X												代表触发器级别名称
{TRIGGER.STATUS}	X												代表触发器状态，可能的值为OK或PROBLEM
{TRIGGER.TEMPLATE.NAME}	X												代表定义了指定触发器的模板列表，如果指定触发器是在主机上被定义的，则该宏变量的值为“UNKNOWN”
{TRIGGER.URL}	X												代表触发器的URL
{TRIGGER.VALUE}	X							X					代表触发器的当前值。如果触发器的状态为OK，则该宏变量的值为0；如果触发器的状态为PROBLEM，则该宏变量的值为1；而如果触发器的状态为UNKNOWN，则该宏变量的值为2。该宏变量也可以使用在触发器表达式中
{TRIGGERS.UNACK}						X							代表与拓扑图中元素相关的，未被确认的触发器的数量，不区分触发器的状态。某个触发器只要有一个与之相关的问题事件未被确认，则该触发器即被视为未确认
{TRIGGERS.PROBLEM.UNACK}						X							代表与拓扑图中元素相关的，未被确认的问题触发器

续表

宏变量	1	2	3	4	5	6	7	8	9	10	11	12	描述
													的数量。某个触发器只要有一个与之相关的问题事件未被确认，则该触发器即被视为未确认。该宏变量与上面所述宏变量的区别在于，该宏变量只统计仍处于问题状态下的触发器的数量
{TRIGGERS.ACK}						X							代表与拓扑图中元素相关的，已被确认的触发器的数量，不区分触发器的状态。某个触发器只要有一个与之相关的问题事件未被确认，则该触发器即被视为未确认
{TRIGGERS.PROBLEM.ACK}						X							代表与拓扑图中元素相关的，已被确认的问题触发器的数量。某个触发器只要有一个与之相关的问题事件未被确认，则该触发器即被视为未确认。该宏变量与上面所述宏变量的区别在于，该宏变量只统计仍处于问题状态下的触发器的数量
{host:key.func(param)}	X					X ³		X					应用在触发器表达式中的简单宏变量
{\$MACRO}					X			X	X	X	X	X	用户自定义的宏变量

备注：

- 【1】 自从 Zabbix 2.0.3 版本开始，在监控项目关键字中出现的 HOST.*形式的宏变量，只有在那些需要使用监控接口的监控项目中，该类宏变量才能正常工作。因此，如果这类宏变量出现在“Zabbix Agent（主动方式）”、“计算”等类型的监控项目关键字中，它将不能正常工作。
- 【2】 依据上下文的不同，该宏变量所代表的值可能也不同。
- 【3】 在拓扑图的标签中，该宏变量仅支持 last、avg、max 和 min 函数，且其函数的参数必须是秒。
- 【4】 对应的宏变量在 Zabbix 2.0.3 版本之后的系统中才开始被支持。
- 【5】 上面我们列出的宏变量并非是 Zabbix 系统所支持的全部宏变量，对于那些在实际工作中极少使用到的宏变量，这里将它们略去了。读者如果需要查看 Zabbix 系统支持的宏变量的完整列表，可参考 Zabbix 官方文档。

参考文献

- [1] <https://www.zabbix.com/documentation/2.0/manual>
- [2] <https://www.zabbix.com/documentation/2.2/manual>
- [3] 雷震甲. 计算机网络管理. 西安: 西安交通大学出版社, 2000
- [4] Baron Schwartz, Peter Zaitsev, Vadim Tkachenko. 高性能 MySQL. 宁海元, 周振兴, 彭立勋, 等, 译. 3 版. 北京: 电子工业出版社, 2013
- [5] http://www.easysoft.com/products/data_access/odbc_odbc_bridge/getting_started.html
- [6] Wesley J Chun. Python 核心编程. 宋吉广译. 2 版. 北京: 人民邮电出版社, 2008
- [7] Eduardo Ciliendo Takechilka Kunimasa, Linux Performance and Tuning Guidelines(First Edition), International Business Machines Corporation, July 2007
- [8] <http://blog.zabbix.com/scalable-zabbix-lessons-on-hitting-9400-nvps/2615/#comment-84830>
- [9] <http://blog.zabbix.com/reloading-configuration-cache/528/>
- [10] <http://blog.zabbix.com/monitoring-how-busy-zabbix-processes-are/457/>
- [11] http://www.zabbix.com/img/zabconf2011/presentations/Alexei_Vladishev_-_Zabbix_Performance_Tuning.pdf
- [12] <http://www.gnu.org/software/gettext/manual/gettext.html>
- [13] http://cn2.php.net/get/php_manual_zh.chm/from/this/mirror
- [14] <http://caiguanguang.blog.51cto.com/1652935/1377089>
- [15] http://www.yddz1718.com/yddz1718_Article_16608.html
- [16] <http://www.ttlsa.com/zabbix/zabbix-agent-types-and-all-keys/>
- [17] <http://fossies.org/dox/zabbix-2.2.5/>

Zabbix监控系统

一个人干不过一个团队，一个团队干不过一个系统。有一套好的监控系统软件，可以让你高枕无忧。本书所介绍的这款开源监控系统——Zabbix监控系统，它不但功能强大、输出的数据图形美观，而且操作和管理都非常简单方便。要理解和掌握一款监控系统，仅仅熟练地掌握其日常操作是远远不够的，还需要掌握大量的周边知识。

我与本书的作者王余应相识于15年前，那时候互联网在中国还是一个十分新鲜的事物，王余应作为我团队里的一员骨干力量，从那时起即担当公司服务器的日常监控和管理工作。至此他在系统管理和监控这一领域至少摸爬滚打了十几年。十多年运维的从业经历，使他积累了非常丰富的一线运维经验和宽广的知识面，使他不仅能将Zabbix这一开源的监控系统讲解透彻，而且还穿插讲解了许多与系统监控有关的周边知识，而这也是本书的一大特色。

CareFusion亚太IT总监，汤国忠



博文视点Broadview



@博文视点Broadview

上架建议：计算机 / 监控运维

ISBN 978-7-121-25682-0



9 787121 256820 >

定价：59.00元



责任编辑：张月萍
封面设计：李玲

版权注意事项：

- 1、书籍版权归作者和出版社所有
- 2、本PDF仅限用于个人获取知识，进行私底下的知识交流
- 3、PDF获得者不得在互联网上以任何目的进行传播
- 4、如觉得书籍内容很赞，请购买正版实体书，支持作者
- 5、请于下载PDF后24小时内删除本PDF。